Received 22 September 2023, accepted 27 November 2023, date of publication 5 December 2023, date of current version 22 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3340140

RESEARCH ARTICLE

Improving Software Effort Estimation Models Using Grey Wolf Optimization Algorithm

NADA MOHAMMED ALSHEIKH AND NABIL MOHAMMED MUNASSAR
Faculty of Computers and Information Technology, University of Science and Technology, Aden, Yemen

Corresponding author: Nada Mohammed Alsheikh (nadaalsheikh3@gmail.com)

ABSTRACT One of the Software Development Life Cycle phases is planning the software project. Estimating the software effort is another task in this project planning phase. Software effort estimation is the method of determining how many workers are required to create a software project. Many researchers have focused on this field to increase the precision of software effort estimation and used both algorithmic and non-algorithmic techniques. The most widely used method is the Constructive Cost Model (COCOMO). However, the COCOMO model has a limitation related to the precision of the software effort estimation. Meta-heuristic algorithms are preferred with parameter optimization because they can provide nearly optimal solutions at a reasonable cost. This study aims to enhance the precision of effort estimation by modifying the three COCOMO-based models' coefficients and assess the efficiency of Grey Wolf Optimization (GWO) in finding the optimal value of effort estimation through applying four other algorithms, including Zebra Optimization (ZOA), Moth-Flame Optimization (MFO), Prairie Dog Optimization (PDO), and White Shark Optimization (WSO) with NASA18 dataset. These models include the basic COCOMO model, and another two models were also suggested in the published research as a modification of the basic COCOMO model. The six most used software effort estimation metrics are used to assess the performance of the proposed models. The results show high accuracy and significant error minimization of the GWO over other algorithms involving ZOA, MFO, PDO, WSO, and other existing models.

INDEX TERMS COCOMO, Grey Wolf Optimization, software effort estimation, software cost estimation, Moth-Flame Optimization, NASA18 dataset, Prairie Dog Optimization, White Shark Optimization, Zebra Optimization.

I. INTRODUCTION

The early stage of software cost estimation of software project development is the most challenging and least accurate task [1]. Software cost estimation is a hot topic and gained attention through continuous research. Researchers concentrate on developing a helpful model that accurately estimates software cost estimation. This has, in turn, led to the development of numerous software cost-estimating models. In general, these techniques are divided into algorithmic and non-algorithmic techniques. The algorithmic methods depend on mathematic formalities to estimate the effort, such as COCOMO, SLM (Software Life Cycle), Function Point, Use Case Point analysis, and Putnam's Model [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Liu.

Non-algorithmic techniques, on the other hand, depend on computation from previous similar project experiences, such as Analog Techniques, Expert Judgment, Parkinson's Law, and Pricing to Win [3].

Software has become a system's primary expense due to the increasing cost or effort associated with software development. The National Aeronautics and Space Administration (NASA) and the Air Force have estimated that the cost of software development can exceed 50% of the overall expenditure due to the highly developed systems used in NASA software projects, both in terms of hardware and software [4]. This has contributed considerably to designing a new model to estimate software effort. There are several precise models for estimating software effort; however, there is still a dire need for more accurate models.

The Constructive Cost Model (COCOMO) is one of the effective methods that measures a project's early-stage effort. This may help reduce the project's overall cost [5]. It has gained considerable popularity, and its flexibility covers a wide range of factors and may be applied in various environments. However, the COCOMO model has a limitation related to the precision of the software effort estimation due to the problem's deterministic nature, and it has coefficients that affect its accuracy. The coefficients' values are fixed for projects of the same kind, but these parameters differ between organizations.

Therefore, fine-tuning the coefficients is required to achieve accurate estimation. Additionally, the meta-heuristic algorithms improve the COCOMO's parameters, which are regarded as successful in accurately estimating the effort of software projects because they rely on population-based search. They also include some equations that use local search to converge to the solution rather than global search to prevent local optimum. This makes it successful in optimizing parameters and feature selection, such as GWO, ZOA, MFO, PDO, and WSO.

By referring to the advantages of GWO, this study suggests that the Grey Wolf Optimization (GWO) algorithm optimizes the parameter values for the three COCOMO-based models, including the basic COCOMO model, Sheta's Model 1 (also called Model I), and Sheta's Model 2 (also called Model II). To assess the efficiency of GWO in finding the optimal value of effort estimation, four other algorithms are applied, including Zebra Optimization (ZOA), Moth-Flame Optimization (MFO), Prairie Dog Optimization (PDO), White Shark Optimization (WSO), and other existing models.

The proposed models are trained with the Nasa18 projects dataset, which represents 70% of the dataset for training and 30% for testing. To assess the performance of the proposed models, the six most employed software effort estimation metrics are used, including VAF, MSE, MMRE, MAE, RMSE, and R^2.

The motivation for this work is that in software project management, approximately 65% of projects fail because of management factors. In addition, the most important one of these factors is inaccurate estimation. To get an accurate software effort estimation, it is necessary to obtain an accurate prediction method. It is highly expected to get an accurate effort estimation. Still, no such model can predict the effort to develop software projects due to the uncertainties and imprecision associated with the software development process. Due to this real-life optimization problem and because of its importance in successful or unsuccessful software projects.

Most researchers focus their work on COCOMO to develop software effort estimation. Even though many researchers have done software effort estimation, none of them can achieve a satisfactory result that can help the software product deliver on time, on budget, and as per the requested quality. This research aims to improve the accuracy of software effort estimation by optimizing the coefficients of three COCOMO-based models. These models include the basic COCOMO model and another two models that have been suggested in the literature as extensions of the basic COCOMO model, which are called Sheta's Model I, and Sheta's Model II using meta-heuristic algorithms, including (GWO), (ZOA), (MFO), (PDO), and (WSO).

The limitation of this study is that the estimation process is not beyond the system level. In other words, a software project in the COCOMO is considered a homogeneous entity composed of a single sub-system. However, in reality, a software system may composed of smaller and heterogeneous sub-systems. Moreover, the study had a limited sample size, but it could improved by increasing it. Also, the study does not consider certain factors for software cost estimation, such as hardware, personal quality, experiences, and tools.

This work explores a study of how the Gray Wolf Optimization algorithm (GWO) enhances the software effort estimation process overall. Where the significant contributions are:

- Proofing the convenience of the Gray Wolf Optimization algorithm can generate general prediction models in the field of software effort estimation.
- The significant improvement in performance over the pre-existing models.
- The machine learning approach is a convenient software effort prediction using a small number of dataset projects and input variables.

This paper is organized as follows: section II describes some related works, section III presents the dataset and the evaluation metrics, section IV explains software effort estimation models, section V interprets the utilized meta-heuristic algorithms, section VI introduces the proposed method, section VII provides the finding, section VIII provides the conclusion and future work.

II. RELATED WORK

The most critical task in developing software is to estimate the cost correctly because any estimation error might result in either an overestimate or an underestimate, which could impact the project's resources. This section presents previous studies and methods previously used for effort estimation and improving the existing models' coefficients in a similar field. Examples of these techniques are GA [6], HACO-BA [7], HWA [8], FPA [9], and eDTO [10].

Almost three decades ago, when Boehm introduced COCOMO [11], several academics presented various cost-estimating models to address a variety of optimization problems. The extensive spread may be noticed in the field of optimizing the basic COCOMO Model [4], [5], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38].

In 2006, Sheta introduced two models, Sheta's Model I and Sheta's Model II, as extensions of the basic COCOMO Model. Researchers moved to develop models because the parameters proposed by Sheta improved the accuracy and quality of the basic model [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50]. However, those researches were excluded form comparison with our study because those researches were not under equal conditions; the differences in the kind of datasets and how to divide the dataset into training and testing, the evaluation metrics, fitness functions, the platforms that were used, and the set of parameters, such as the number of iterations, population size… etc.

Verma et al. employed GA to optimize the intermediate COCOMO model's coefficients and used the NASA dataset to achieve the results. They carried out their study by utilizing the model's organic mode. The fitness function is Manhattan Distance (MD). The applied methodology yielded better results than the intermediate COCOMO coefficients [6].

Khan et al. analyzed the algorithms from two angles. The first is to compare the efficiency of the Ant Colony Optimization (ACO), the Bat Algorithm (BA), and the Hybrid of Ant Colony Optimization with the Bat Algorithm (HACO-BA) in improving COCOMO II's parameters. The second is that the authors used HACO-BA to enhance the deep learning training process to minimize the training delays. The results confirmed the superiority of the HACO-BA over BA and ACO. Moreover, HACO-BA performed more effectively when DNN was optimized for execution speed and precision [7].

Chhabra et al. suggested two models. Firstly, the fuzzy techniques create a fuzzy technique for each cost driver, and secondly, they use a Genetic Algorithm to select parameters, characterizing fuzzy sets for the proposed fuzzy technique. COCOMO NASA and COCOMO NASA 2 datasets are used to validate the proposed model. The results are compared to the values that match the COCOMO model, and the proposed GA-optimized fuzzy COCOMO provided the best result with reduced MMRE and increased Pred (25%) level [51].

Fadhil and Alsarraj applied the Humpback Whale Algorithm on COCOMO II to find the optimal coefficient values. The authors used NASA93. For performance evaluation, they used MRE and MMRE. A comparison is also conducted between the proposed model and previous similar completed models, including Hybrid Cuckoo Optimization, Harmony Search Algorithm, and CSHS. The results showed that the Whale Algorithm outperformed COCOMO II and CSHS, whereby MMRE is 57.40 for COCOMO II, 54.04 for CSHS, and 50.6122 for the Whale Algorithm [8].

Suherman et al. attempted to verify whether the tuning parameters, which utilize machine learning, such as Random Forest Regression, can outperform the constant parameters of COCOMO II with accurate results or vice versa. The proposed method is implemented on NASA 93. Their results showed that Random Forest Regression outperformed SVR and Bee Colony [52].

Puspaningrum et al. applied the Flower Pollination Algorithm in several iterations: 500, 1000, 1500, 2000, and 2500. The authors used 500 iterations to compare with COCOMO II, Cuckoo Search Algorithm, and PSO. Besides, they used NASA 93, and the results showed the superiority of the Flower Pollination Algorithm versus other algorithms, obtaining 5248% [9].

Shweta et al. applied the Ensemble Duck Traveler Optimization Algorithm (eDTO) to enhance COCOMO II's accuracy in estimating the time and effort required to develop software and assess the performance of the proposed model. The authors used NASA93 and evaluation metrics ACC, VAR, BRE, MRE, and MMRE to compare the results with the existing COCOMO II model, Neural Network, and Strawberry Algorithm. The results verified the superiority of eDTO over other algorithms in all evaluation metrics [10].

Two models are proposed by Fadhil et al. to optimize the parameters of COCOMO II: firstly, by using the Dolphin Algorithm, and secondly, by employing the hybrid Dolphin and Bat Algorithm (DOLBAT) on two datasets, NASA93 and NASA60. To carry out the performance evaluation, the authors used MRE and MMRE. The results showed that DOLBAT outperformed GA, CSHS, Bat, and Dolphin. For NASA93, the MMRE of the DOLBAT is 50.27, while for the Dolphin, it is 51.8755. Moreover, for NASA60, the MMRE of the DOLBAT is 14.57, while for the Dolphin, it is 16.65 [53].

Fadhil and Bahnam proposed a hybrid model of the Ant-Lion Optimization and Cuttlefish Algorithms (HALOCF) to improve the COCOMO II's parameters. The authors used two datasets, including NASA 93 and NASA 60. For evaluation, they used the MMRE metric. They compared the results with several algorithms, such as COCOMO II, CSHS, Bat, ALO, and CF. The results demonstrated that the HALOCF outperformed compared with other algorithms in MMRE in two datasets [54].

Saleem et al. proposed a systematic review to discuss several software cost estimation techniques and identified their strengths and weaknesses. The authors focused on understanding issues that cause cost estimation problems in software projects. They concluded that some models performed better in large projects, others in small projects, and some on global software. Therefore, to obtain the best efficiency, the authors recommended using the collaborative methods [55].

Khan et al. proposed the Flower Pollination Algorithm (FPA) to optimize intermediate COCOMO's coefficients. They used NASA93, NASA 63, and NASA 60 datasets. It was found that the FPA has effectively solved this optimization problem and obtained the best results when compared to the normal value of COCOMO's parameters. The results showed that the improvement on NASA93 is 10.17%, NASA 63 is 77.38%, and NASA 60 is 22.96% [56].

Sethy et al. employed the TLBO algorithm to increase the precision of the COCOMO model by refining the COCOMO's coefficients. They used the IVR dataset, which includes 47 projects. The authors compared the results with previous models, such as Bailey, COCOMO 2, Hastead, SEl, and BCO. The TLBO algorithm provided high accuracy by obtaining the lowest value of MMRE [57].

Ullah et al. suggested a Biogeography-Based Optimization (BBO) model to improve the existing coefficients of COCOMO II. They used two datasets, NASA93 and Turkish industry software projects evaluated by MD and MMRE. The results demonstrated that the proposed model has significant precision and reduces error compared to COCOMO II, PSO, FPA, GA, and other high-cost estimation models [58].

Sunindyo and Rudiyanto suggested combining COCOMO II and the K-Means clustering approach to increase the precision of the COCOMO II by determining new values of a and b. They used the COCOMO NASA2 dataset and the Turkish Software Industry. The results showed that the proposed model could reduce the amount of MRE COCOMO II from 1.32 to 0.85 and improve the amount of PRED (0.3) from 32% to 54% [59].

Singal et al. applied the Differential Evolution Algorithms to optimize the COCOMO and COCOMO II parameter values. The parameter values are obtained by using three mutation techniques in Differential Evolution. The proposed model is tested on the COCOMO 81 and NASA93 datasets. The test results showed the proposed model's superiority on the COCOMO and COCOMO II while obtaining the lowest MMRE for the two datasets [60].

Singh et al. suggested an Enhance-Based Differential Evaluation Algorithm (EABMO) to tune the parameters of the semidetached model. The performance of the developed model was tested on NASA and compared to other modern DE algorithms, PSO, and GA. The result showed that the proposed EABMO algorithm outperformed DE, PSO, and GA [61].

Vats et al. proposed a Genetic Algorithm to develop a cost estimation framework using an object-oriented perspective and a prediction model using regression techniques and GA. The evaluation metrics used RMSE and MAE. The result showed that GA outperformed on regression in the RMSE 96.31 for regression, GA is 61.66, MAE 0.17188, and 0.098818 for GA [62].

Reddy and Behera surveyed to study the benefit of PSO in software estimation. They analyzed the previous works. They concluded that the datasets most frequently used are COCOMO81, NASA, Maxwell, and ISBSG. They used the evaluation metrics that most frequently applied MMRE, PRED, MARE, and MDMRE [63].

Kumar and Behera proposed several machine learning methods, such as KNN, SVM, Neural Networks, Random Forests, and Backpropagation Algorithms, for estimating software effort estimation. They used COCOMO81 and Desharnais. They divided the dataset into 80% for training and 20% for testing. The orange tool for comparing. The results, it is found that the software effort estimation using KNN is better than other models [64].

The research gaps that were found in previous works:
• Lack of estimation accuracy.
• Their focus was on improving one model.
• They did not use unseen data in the testing phase, which makes the models unable to generalize.
• Use one or at most two metrics to measure the model's performance.
• The experimental result could improve the original COCOMO. However, there are higher relative errors to the actual effort of projects.
• They didn't consider the effect of methodology.

III. DATASET DESCRIPTION AND EVALUATION METRICS

This research examines a famous and public dataset to produce comparable results with [76] under equal conditions, namely the NASA projects' effort dataset. The dataset is challenging due to the few cases and the small number of variables. However, the dataset is considered sufficient for this study's objectives. The dataset for this study is taken from NASA software project data, which is collected by Bailey dataset and Basili [65].

The dataset has 18 software projects; each project has three variables. First, the Kilo Line Of Code (KLOC), which is presented in Kilo Line of Code and chiefly determines the effort of a software project. Second, the Methodology (ME) has a real effect on the cost of a particular software project. The last one is the measured effort, which is described in man-months. It is worth mentioning that 70% of the dataset is used for estimating the model's parameters and 30% for testing their performance. The utilized dataset in this study has been used by many researchers. Table 2 presents the dataset in this study.

In this study, the first 13 projects' data were used to optimize the parameter values, while the remaining 5 projects' data were used to test the models. The evaluation metrics, which are used for calculating the differences, are as follows:

$$\text{``}Variance - Accounted - For\ (VAF)\ \ as\ in\ \ (1)'' \qquad [5]$$

$$VAF = [1 - \frac{var(act - est)}{var(act)}] * 100\ \% \tag{1}$$

$$\text{``}Mean\ \ Squared\ \ Error\ (MSE)\ \ as\ in\ \ (2)'' \qquad [28]$$

$$MSE = 1/n \sum_{i=1}^{n} (act - est)^2 \tag{2}$$

$$\text{``}Mean\ of\ Absolute\ Error\ (MAE)\ \ as\ in\ \ (3)'' \qquad [39]$$

$$MAE = 1/n \sum_{i=1}^{n} |act - est| \tag{3}$$

$$\text{``}Mean\ Magnitude\ Relative\ Error(MMRE) as\ in (4)'' \qquad [5]$$

$$MMRE = 1/n \sum_{i=1}^{n} \frac{|act - est|}{act} \tag{4}$$

$$Root\ Mean\ Squared\ Error\ (RMSE)\ \ as\ in\ \ (5)'' \qquad [66]$$

TABLE 1. Summary of related works.

Reference	Algorithm used	Dataset used	Evaluation metric	Model used	Fitness function used
[12]	Nature-Inspired Algorithm(NIA)	Nasa 18	MAE	Basic COCOMO	MAE
[39]	Directed Artificial Bee Colony Algorithm	Nasa 18	MMRE, MRE,and PRED(N)	Basic COCOMO, Sheta's model 1, and Sheta's model 2	MRE
[13]	Strawberry Plant	Nasa 93	MMRE and MRE	Basic COCOMO	MMRE
[4]	Differential Evaluation (DE)	Nasa 18	MMRE and VAF	Basic COCOMO	VAF
[14]	Simplified GA	Nasa 18	MMRE, MD, and VAF	Basic COCOMO	MD
[39]	Directed Artificial Bee Colony Algorithm	Nasa 18	MMRE, MRE, and PRED(N)	Basic COCOMO, Sheta's model 1, and Sheta's model 2	MRE
[33]	Fuzzy Logic technique	Nasa 18	MMRE and RMSE	Basic COCOMO	MRE
[37]	Bee Colony Algorithm	IVR dataset	MMRE and RMSE	Basic COCOMO	MRE
[34]	Genetic Programming (GP)	Nasa 18	VAF and MMRE	Basic COCOMO	MRE
[41]	Genetic Algorithm (GA)	Nasa 18	MMRE, VAF, MRE, and PRED(N)	Basic COCOMO	MRE
[42]	Genetic Algorithm (GA)	Nasa 18	MMRE, MdMRE, MRE, and PRED(N)	Basic COCOMO, Sheta's model 1, and Sheta's model 2	MRE
[43]	Particle Swarm Optimization(PSO)	COCOMO81	MMRE, MRE, and VAF	Basic COCOMO, Sheta's model 1, and Sheta's model 2	MRE
[26]	Human Opinion Dynamic-COCOMO (HOD-COCOMO)	Nasa 93	MMRE	Basic COCOMO	MMRE
[28]	Teaching-Learning based optimization Algorithm	Nasa 18	MRE	Basic COCOMO	MRE
[29]	Greyrelation Analysis (GRA) and Bat Algorithm	Kermer dataset	MMRE and MRE	Basic COCOMO	MRE
[30]	Cuckoo Search (CS) Algorithm	Nasa 18	MMRE and PRED(N)	Basic COCOMO	MRE
[32]	Bacterial Foraging optimization Algorithm (BFOA)	Martins Dataset	MMRE,MRE and PRED(N)	Basic COCOMO	MMRE
[17]	Bat insprid Gravitational Search (BATGSA)	Nasa 60, Nasa 93, COCOMO 81, and Kermer	MAE and NE	Basic COCOMO	MAE
[18]	Harmony Search Algorithm	Nasa 93	MMRE	Basic COCOMO	MMRE
[44]	Particle Swarm Optimization (PSO)	Nasa 18	ARE, MARE, and VAF	Basic COCOMO, Sheta's model 1, and Sheta's model 2	VAF

$$RMSE = \sqrt{\sum_{i=1}^{n}(act - est)^2} \tag{5}$$

$“R - Squared\ (R^2)\ \ as\ in\ (6)''$ [28]

$$R^2 = \frac{\sum_{i=1}^{n}(act - mean(act))^2 - \sum_{i=1}^{n}(act - est)^2}{\sum_{i=1}^{n}(act - mean(act))^2} \tag{6}$$

IV. SOFTWARE EFFORT ESTIMATION MODELS

Software effort estimation is the method of determining how many workers are required to create a software project. The Constructive Cost Models COCOMO and its modification COCOMO II are well-known and often-used effort estimation models. COCOMO, sometimes known as COCOMO 81, is used as a cost, effort, and schedule

TABLE 2. NASA 18 software projects.

Project No.	KLOC	Methodology (ME)	Measured Effort
1	90.2	30	115.8
2	46.2	20	96
3	46.5	19	79
4	54.5	20	90.8
5	31.1	35	39.6
6	67.5	29	98.4
7	12.8	26	18.9
8	10.5	34	10.3
9	21.5	31	28.5
10	3.1	26	7
11	4.2	19	9
12	7.8	31	7.3
13	2.1	28	5
14	5	29	8.4
15	78.6	35	98.7
16	9.7	27	15.6
17	12.5	27	23.9
18	100.8	34	138.3

estimation model for planning new software development activities. The COCOMO was defined in 1981 [11].

Meanwhile, COCOMO II is a later extension of the model that was initially established. In this study, the three COCOMO-based model coefficients are being optimized. The first is the basic COCOMO, represented in Equation (7). The other two models are modifications of the basic COCOMO model, which are the proposed models by Sheta [67], which are based on the Boehm Constructive Cost Model(COCOMO), are tuned in this work. Sheta made some adjustments to Boehm's basic model for software cost estimation to obtain a generalized model for estimating the effort for all types of projects. He applied a Genetic Algorithm to tune the parameters of the proposed models to forecast the precise estimation of effort.

Sheta has also added methodology to the COCOMO to increase the prediction accuracy and quality of the COCOMO model. He has also discovered that adding a bias term comparable to the regression model classes stabilizes the model and lessons the impact of measurement noise. The models created by Sheta [67] were successful in enhancing the estimated effort's performance relative to the VAF criteria.

A and B are parameters in the basic model in Equation (7). Other parameters, i.e., C and D, were added to the basic model to improve the prediction accuracy, as shown in Equations (8,9). Sheta's Model 1, which is called Model I, and Sheta's Model 2, which is called Model II, are given below:

$$E = A(KLOC)^B \tag{7}$$

$$E = A(KLOC)^B + C(ME) \tag{8}$$

$$E = A(KLOC)^B + C(ME) + D \tag{9}$$

Here, E denotes the estimated effort, A and B denote multiplicative and exponential constant, respectively, KLOC is the software project's size, and ME is the methodology.

A. SOFTWARE COST ESTIMATION STEPS

According to software engineering economics, estimating the cost of a software project includes four steps [11]. Each step gives an input to the next one. Software size is used as an input in effort estimation along with other attributes that influence the software effort needed to develop software. The software size and effort estimation are used to determine the calendar time or schedule that was required to construct the software. The steps to determine the cost of a particular software project are as follows:

- Software size estimation.
- Software effort estimation.
- Software time estimation.
- Software dollar cost estimation.

1) SOFTWARE SIZE ESTIMATION

The most crucial software management task is size estimation, the first step in software engineering economics that calculates the software project effort. Due to subsequent work, estimating effort and time depends on software size. The project manager must be aware of the project size of a particular software project to calculate the software cost and identify how many people should be allocated.

Lawrence H. Putnam LOC, the Line Of Code and Function Point Analysis was used to estimate the line of code.

a: LAWERECE H.PUTNAM LOC ESTIMATION

In this method, the Line Of Code is estimated by dividing the system into smaller parts and calculating the SLOC of each one. In this approach, for each part of the software system, the smallest piece SLOC, most likely SLOC, and the largest possible SLOC estimates are made by up to three to four specialists for part of a software system. Then, the predicted SLOC is calculated for each part of the software system using Equation (10).

$$Ei = \frac{a + 4m + b}{6} \tag{10}$$

where a represents the smallest possible SLOC, b represents the largest possible SLOC, and m is most likely SLOC, respectively. Then, the expected software size for the SOLC of the entire system and computed by the following Equation (11).

$$Ei = \sum\nolimits_{i=1}^{n} Ei \tag{11}$$

where n, represents the total number of parts in the whole system.

b: FUNCTION POINT ANALYSIS

According to this method, the software size is calculated using units. Counting the number of external components(such as inputs, outputs, inquiries, and interfaces) that the system is made up. The system's input files, tables, forms, screens, and messages will be counted as a component of software size from the external inputs. External I/O inquiries that demand a response, such as prompts, interruptions, calls, etc.…, are counted. Software size is also affected by libraries or programs that are passed through and out of the system. The following steps are being taken to assess the software size of a project:

1. Estimate or count each external type's occurrences (inputs, outputs, inquiries, and interfaces).
2. Each occurrence should be given a complexity weight.
3. To calculate the function count, multiply each occurrence by its complexity weight, then add the results.
4. To determine the function point count, multiply the function count by a value adjustment multiplier (VAM).

$$VAM = \sum vi * 0.01 + 0.065 \tag{12}$$

The following table can multiply each occurrence by its complexity weight.

TABLE 3. Complexity weight.

Description	Low	Medium	High
Externals inputs	3	4	6
Externals outputs	4	5	7
Externals inquiries	3	4	6
Externals interfaces	5	7	10
Internal data files	7	10	15

2) SOFTWARE EFFORT ESTIMATION TECHNIQUES

Software effort and cost estimation techniques are divided into algorithmic and non-algorithmic techniques. The algorithmic methods depend on mathematical formalities to estimate the effort, such as the Constructive Cost Model (COCOMO), SLM (software life cycle), Function Point, Use Case Point analysis, and Putnam's Model [2].

a: CONSTRUCTIVE COST MODEL (COCOMO)

The Constructive Cost Model (COCOMO), which Barry Boehm proposed in 1981 based on an analysis of 63 projects, is one of the most significant, well-documented, commonly used algorithmic models [11]. Using the software's size and other cost drivers, this model calculates the cost and effort of the software. The basic COCOMO Model, intermediate COCOMO Model, and advanced COCOMO Model are the three basic variations of the model. The COCOMO model measures effort in terms of person-months and measures code size in lines of code (LOC) or thousand lines of code (KLOC).

i) BASIC COCOMO MODEL

The basic Constructve Cost Model needs the software size estimated by Line Of Code or Function Point Analysis. The project types in this model are organic, semidetached, and embedded. Its classification is mainly based on the project's size, which is the organic less than 50 KLOC, the semidetached is 50-300 for KLOC, and the embedded is over 300 KLOC, represented by Equation (7).

TABLE 4. Coefficients value in basic COCOMO model.

Basic-COCOMO projects	A	B
organic	2.4	1.05
semidetached	3.0	1.12
embedded	3.6	1.20

ii) INTERMEDIATE-COCOMO MODEL

The intermediate-COCOMO model includes cost drives and the line of code used in the basic COCOMO model. Cost drives consist of products, personnel, hardware, and projects. So, the estimated cost and effort combine the line of code and the cost drives. In the intermediate-COCOMO model, nominal effort estimation is calculated using the power function of A and B, with the value slightly different from the basic COCOMO model. The cost factors range from 0.7 to 1.66, and the estimated effort is calculated using Equation (7).

$$E = A(KLOC)^B \times EMF \tag{13}$$

where EMF, is the product of all the cost factors.

TABLE 5. Coefficients value in intermediate COCOMO model.

INTERMEDIATE-COCOMO projects	A	B
organic	3.2	1.05
semidetached	3.0	1.12
embedded	2.8	1.20

iii) DETAILED COCOMO MODEL

The intermediate-COCOMO version's features are included in the detailed COCOMO, along with an evaluation of how the cost driver affects each phase of the software engineering process. The detailed COCOMO uses various effort multipliers, and we use COCOMO in each module to estimate work before adding it all up.

The stages of software development used in the detailed COCOMO Model to calculate software effort are detailed design (DD), code and unit testing (CUT), requirement design and product design (RPD), and integration and test (IT). Each software module's estimated effort determines the effort of the entire system. Table 7 shows the rating scale for each cost driver in the four phases of the detailed COCOMO Model.

3) COCOMO II MODEL

The COCOMO Model was created based on the waterfall software development process paradigm. COCOMO II was designed to incorporate the most recent model for the

TABLE 6. Cost factor and their weight in intermediate COCOMO.

Effort Multipliers Code	Multipliers name	Rating					
	Personnel Attributes	Verylow	Low	Nominal	High	Very high	Extra high
ACAP	Analyst capability	1.46	1.19	1.00	0.86	0.71	
AEXP	Application experience	1.29	1.13	1.00	0.91	0.82	
PCAP	Programmer capability	1.42	1.17	1.00	0.86	0.70	
VEXP	Virtual Machine experience	1.21	1.10	1.00	0.90	-	
LEXP	Language experience	1.14	1.07	1.00	0.95	-	
	Project attributes						
MODP	Model Programming practice	1.24	1.10	1.00	0.91	0.82	
TOOL	Development schedule	1.24	1.10	1.00	0.91	0.83	
SCED	Software tools	1.23	1.08	1.00	1.04	1.10	
	Product attributes						
RELY	Required Software reliability	0.75	0.88	1.00	1.15	1.40	
DATA	Database size	-	0.94	1.00	1.08	1.16	
CPLX	Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
	Computer attributes						
TIME	Execution Time constraint	-	-	1.00	1.11	1.30	1.66
STOR	Main storage constraint	-	-	1.00	1.06	1.21	1.56
VIRT	Virtual Machine volatility	-	0.87	1.00	1.15	1.30	
TURN	Computer Turnaround time	-	0.87	1.00	1.07	1.15	

software development process. The COCOMO II model can be used to determine the effort of a software project when the development process is incremental, iterative, or spiral or when re-engineering is necessary. Equation (14) determines a project's effort during either the early design phase or post-architecture. The unit of measurement for effort is Person-Month (PM). Person Month is the time that one person works on the software project development for one month.

$$PM = A \times Size^{E} \times \prod_{i=2}^{n} \times EMF \tag{14}$$

$$E = B + 0.01 \times \sum_{j=0}^{5} \times SF \tag{15}$$

TABLE 7. Effort multiplier rating scale and its value for detailed COCOMO model.

Rating	RPD	DD	CUT	IT
Very low	1.80	1.35	0.0287	1.50
Low	0.85	0.85	0.85	1.20
Nominal	1.00	1.00	1.00	1.00
High	0.75	0.90	0.90	0.85
Very high	0.55	0.75	0.75	0.70

where n, represents the number of effort multiplier in the early design or post-architecture, n is 17 for post-architecture, and 7 for the early design model. SF represents the five scale factors in COCOMO II. A and B are constants whose value is derived from 161 software projects. EM is the product of 17 effort multipliers. The five scale factors in the COCOMO II are precedentedness (PREC), development flexibility (FLEX), risk resolution (RESL), team cohesion (TEAM), and process maturity (PMAT), and there are 17 of them. The COCOMO II effort multipliers are represented in Table 8, along with their corresponding values.

B. SLIM MODEL

Software Life Cycle Management is one algorithmic model used for large projects. It is sometimes referred to as a macro estimation model. It was one of the first models of empirical and algorithmic software cost estimation. The method describes the time and software effort required to construct a software project. Using Equation (16), the software effort is estimated.

$$Effort = [\frac{Size}{(productivity \times Time)(\frac{4}{3})}] \times B \tag{16}$$

where Size, is the estimated size of the software product, productivity is the productivity of the organizational process.

C. EXPERIENCE BASED ESTIMATION

The estimation technique for software projects is commonly utilized when gathering requirements and data is challenging. The estimation is calculated using residents' experiences.

D. ESTIMATION BY ANALOGY

The technique of developing a solution using similar tasks that have already been done and applying that answer to a new problem area is known as estimation measurement. The analogy technique is identical to experience-based estimation, except it only uses prior projects' knowledge and experience. While estimating by analogy is a data-intensive strategy based on one or more identified prospective similar projects, experience-based estimation is a human-intensive approach.

E. TOP DOWN AND BOTTOM UP APPROACH

The estimated total effort is either determined as the sum of the project activity estimates or is based on the characteristics of the project as a whole. The accuracy with which algorithmic methods and experience-based estimation techniques may estimate the effort value of software projects is limited. The use of meta-heuristic algorithms in estimating the time and cost of software projects is currently prevalent and results in better estimates.

V. THE META-HEURISTIC ALGORITHMS

A. GREY WOLF OPTIMIZATION ALGORITHM

The GWO algorithm is mainly motivated by the gray wolves' social hierarchy and hunting method. The social hierarchy is separated into four levels. The alpha (α), beta (β), delta (δ), and omega (ω).

The alphas are leaders who may be females or males. They are not necessarily the strongest, but they can best manage the herd. They are responsible for making orders regarding hunting, where to sleep, when to wake up, and so on. The alpha wolf is known as the dominant wolf because the rest of the herd must follow the orders it gives.

The beta is the second level of the hierarchy; it is an advisor to the alpha and helps the alpha make decisions, and it takes orders from the alpha and gives orders to the rest of the herd. If the alpha wolf dies, it becomes the first candidate to be the herd leader.

The third level is the delta wolf; it obeys the alpha and beta commands and controls the omega. They are the scouts, elders, and caretakers who look after the herd's injured and weak wolves.

The lowest rank is the omega wolf, which plays the role of a scapegoat and must obey all orders of the alpha and beta, and it is the last wolf allowed to eat [73].

The main phases of gray wolves' hunting are as follows:

- Chasing, pursing, and closing in on the prey.
- Chasing the victim until it stops moving, then surrounding and pestering it.
- Attacking the victim.

The computation complexity of the GWO algorithm is defined as follows:

$$O(m(dn + n + nlogn + n) \tag{17}$$

1) THE REASONS FOR CHOOSING THE GWO ALGORITHM ARE

The first reason is that the GWO has proven its efficiency in studies similar to the nature of our research problem, such as in [74], where they proposed a model to predict rainfall based on the input time-series weather data using GWO in India, Jammu, and Kashmir. They used four datasets. The result demonstrated that it outperformed the proposed Accuracy, MSE, and PRD methods.

The second reason is that this study [75] demonstrated that the GWO algorithm has attracted from several sources and is being used in a variety of fields, including job scheduling,

TABLE 8. COCOMO II effort multipliers.

Scale Factors	Very low	Low	Nominal	High	Very high	Extra high
RELY	0.82	0.92	1.00	1.10	1.26	
DATA		0.90	1.00	1.14	1.28	
CPLX	0.73	0.87	1.00	1.17	1.34	1.74
RUSE		0.95	1.00	1.07	1.15	1.24
DOCU	0.81	0.91	1.00	1.11	1.23	1.24
TIME			1.00	1.11	1.29	1.63
STOR			1.00	1.05	1.17	1.46
PVOL		0.87	1.00	1.15	1.30	
ACAP	1.42	1.19	1.00	0.85	0.71	
PCAP	1.34	1.15	1.00	0.88	0.76	
PCON	1.29	1.12	1.00	0.90	0.81	
APEX	1.22	1.10	1.00	0.88	0.81	
PLEX	1.19	1.09	1.00	0.91	0.85	
LTEX	1.20	1.09	1.00	0.91	0.84	
TOOL	1.17	1.09	1.00	0.90	0.78	
SITE	1.22	1.09	1.00	0.93	0.86	
SCED	1.43	1.14	1.00	1.00	1.00	

Surface Waves, Unmanned Combat Ariel Vehicles (UCAVs), Optimum Reactive Power Dispatch (ORPD), Bankruptcy Prediction, and Smart Green House. According to results from applications of the GWO, The GWO performed better than other bio-inspired algorithms, including BA, GA, FA, PSO, and others.

The third reason is that our research problem is a single-objective optimization problem, and the GWO solves single-objective and multi-objective optimization problems. The fourth reason is that according to the advantages of the GWO, which are:

- Ease of implementation.
- Its simplicity because it has the two primary parameters to be adjusted (a and C).
- The social hierarchy assists GWO in saving the fittest solutions obtained so far developed throughout the iteration.

• The encircling mechanism establishes a hyper-spherical neighborhood in higher dimensions around the solutions.

• Candidate solutions are helped to have hyperspheres with various random radii by the random parameters A and C.

• The hunting technique enables possible solutions to identify the likely location of the prey.

• The a and A adaptive values ensure exploration and exploitation.

• The smooth transition between exploration and exploitation is made possible by GWO's values for the parameters a and A.

• As A decreases, exploration takes up half of the iterations, and exploitation takes up the other half.

TABLE 9. Setting the parameters for the GWO algorithm.

Parameter	Value
Maximum iteration	500
Wolves No	100
a	2
t	1

Algorithm 1 Pseudo-code of GWO

```
Initialization of grey wolves Wn where n=1,2,...,N
  According to given upper bound(UB) and lower
  bound (LB) values.
  Initialize a, A and C.
  Evaluation the fitness of all search agents
  Select alpha, beta and delta as:
  $W\alpha$ = best search agent
  $W\beta$ = second best search agent
  $W\delta$ = third best search agent
  Initialize i = 0 and Max-it = Maximum number of
  iterations allowed.
  while $i < Max - it$ do
    for each search agent do
      Update the position of the current search agent
    end
    Update a, A and C.
      Evaluate the fitness of all search agents.
      If any better solution then update $W\alpha$,$W\beta$, and
      $W\delta$
      i=i+1.
end
Stop the process and visualize the first best agent $W\alpha$
```

B. ZEBRA OPTIMIZATION ALGORITHM

The ZOA algorithm was primarily inspired by the social behavior of herds of zebras in the wild. In particular, this mimics zebras' two most important social behaviors: foraging and defense strategies. The ZOA is a population-based algorithm, whereas the population is zebras. Each zebra is a candidate solution in the search space. Zebra can be represented as a vector, and the values inside this vector are the variables of the problem, i.e., the fitness function values. In contrast, the population of zebras is represented as a metric.

The location of every zebra in the search space determines the values of the parameters for the optimization problem; the initial position is assigned randomly. The zebras is compared based on the zebra's fitness function value in the population, and it can represent the best one based on the type of fitness function, whether maximum or minimum.

In ZOA, the population's best member is the pioneer zebra, representing the global best and leading other zebras toward its position.

In each iteration, the location of all zebras is updated, and the best solution is generated each time. The update is carried out depending on two phases: 1) Foraging and 2) Defense strategies.

In the first phase, the members update their position based on simulating the behaviors of zebras when searching for food, and the best member is the pioneer zebra, which leads other zebras to their position in the search space.

In the second phase, the behavior of zebras in defense against predators is used to update the position of zebras in search space. It depends on the type of predator; if it is a lion, then the strategy they use is to escape in a zigzag pattern. If it were a smaller animal like a dog, zebras would gather to confuse and scare it [68].

The computation complexity of the ZOA algorithm is defined as follows:

$$O(N.m.(1 + 2.T)) \qquad (18)$$

1) THE REASONS FOR CHOOSING THE ZOA ALGORITHM ARE

The first reason is that sixty-eight benchmark functions, which include unimodal, high-dimensional multimodal, fixed-dimensional multimodal, CEC2015, and CEC2017 kinds, were used to assess ZOA's performance in solving optimization problems. The optimization findings demonstrated that ZOA can offer the best solutions for cost functions by achieving the proper balance between exploitation and exploration.

Moreover, four engineering design problems were used to examine ZOA's capacity to optimize real-world issues: welded beam, tension/compression spring, pressure vessel, and speed reducer. It was proven that the ZOA algorithm outperformed other known algorithms in most cases, including GWO, TLBO, GA, MPA, PSO, TSA WOA, and GSA [68].

The second reason is that the ZOA algorithm was proposed in 2022, so it was interesting to use it in optimizing parameters for the old models, such as the basic COCOMO model, which was offered in 1981, and the two Sheta's models, which were proposed in 2006.

The third reason is that we searched in ACM, Scholar, IEEE, and Science Direct databases, and this algorithm has yet to be used to estimate software effort.

TABLE 10. Setting the parameters for the ZOA.

Parameter	Value
Maximum iteration	500
Search agents NO	100

C. MOTH-FLAME OPTIMIZATION ALGORITHM

The MFO is a meta-heuristic algorithm, which is a population-based algorithm. It was created by Sayedali in 2015 [69]. The MFO algorithm imitates the behavior of moths naturally through the moths' crosswise movements.

Real moths use a clever way of flying through the darkness to cover large distances in a straight line by keeping a steady and specific degree toward the moon's position as a light source. However, as soon as the moths see the artificial light nearer than the moonlight, they finally converge on it. They keep the same degree before becoming caught in a spiraling motion around it.

The MFO algorithm consists of two elements: moths and flames. Each moth in the MFO algorithm indicates a solution, and the variables in each problem are determined by the positions of the moths in the search space. The moth is regarded as a search agent that gets the search. By updating their position, the moths could be searched in multiterminal spaces.

The moths' number and flames' dimensions are stored in an array to keep each moth's best position. Both moths and flames metrics allow each moth to search the area and update the flame, which is the best position if a better solution is discovered.

The following can be used to present the MFO algorithm generally: MFO is (I, P, T). There are three-tuple estimation methods in the MFO algorithm: the method I randomly initializes the population, and P describes a method for searching for nearby moth solutions until the termination condition is satisfied, where T is a procedure that returns whether or not the termination condition has been met.

The computation complexity of the MFO algorithm is defined as follows:

$$O(t(n^2 + n \times d)) = O(tn^2 + tnd) \quad (19)$$

1) THE REASONS FOR CHOOSING THE MFO ALGORITHM ARE

The first reason is that we searched in ACM, Scholar, IEEE, and Science Direct databases, and this algorithm has yet to be used to estimate software effort. The second reason is that the MFO algorithm has been proven efficient against several algorithms such as SOS, GA, PSO, ABC, and more when used in various fields, including engineering, chemical and medical research, machine Learning, and image processing… etc [70].

The third reason is due to its advantages, which are:

• It is a mechanism for primarily enhancing exploitation; updating positions enables gathering nearby solutions around the flames.

Algorithm 2 Pseudo-code of ZOA

```
Input: the optimization problem information
Set the number of iterations(T) and the number of zebras' population(N).
Initialization of the position of zebras and evaluation of the objective function.
Evaluation the fitness of all search agents
for $t=1:T$ do
    Update pioneer zebra(PZ)
    for $i=1:N$ do
        Phase 1: Foraging behavior
        Calculate new status of ith zebra by
        $x_{ij}^{new,P1} = x_{ij} + r.(PZ_j - I.x_{ij})$ Update the ith zebra by
        $X_i = \begin{cases} x_{ij}^{new,P1}, & F_i^{new,P1} < F_i; \\ X_i, & else, \end{cases}$
        Phase 2: Defense strategies against predators
        if $Ps < 0.5$, $Ps = rand$ then
            Strategy1: against lion (exploitation phase)
            Calculate new status of the zebra using mode S1
        end
        $X_i = \begin{cases} S1 : x_{ij} + R.(2r-1) .(1 - \frac{t}{T}).x_{ij}, & Ps < 0.5; \\ S2 : x_{ij}^{new,P1} = x_{ij} + r .(AZ_j - I.x_{ij}), & else, \end{cases}$
        else
            Strategy2:against other predator (exploration phase)
            Calculate new status of the zebra using mode S(2)
        end
        Update the ith zebra by
        $X_i = \begin{cases} x_{ij}^{new,P2}, & F_i^{new,P2} < F_i; \\ X_i, & else, \end{cases}$
    end
    Save best candidate solution so far
end
Return the best solution obtained by ZOA for given optimization problem
```

• Over several iterations, the adaptive convergence parameter (r) towards the flame leads to increased exploitation surrounding the flames.

• Due to the use of a population of moths by MFO for optimization, local optimal avoidance is highly effective.

• Each moth is given a flame, leading to search space exploration and lowering the possibility that local optima would stop.

• The search space balances exploitation and exploration by reducing the flame count.

• Using the most recent and successful solution so far as the flames, moths might use the promising solutions as their guides.

• To ensure their preservation, only the fittest solutions are saved.

• Because moths constantly update their positions about flames, representing the most promising solutions so far throughout iterations, the convergence of the MFO algorithm is ensured.

• The MFO method can handle complex real-world issues with uncertain and limited search spaces. [69]

Algorithm 3 Pseudo-Code of MFO

```
Initialization parameters of Moth-Flame
Initialization Moth position $M_i$ randomly
for $i = 1 : n$ do
    Calculate the fitness function
end
while $iteration < Max - iteration$ do
    Update the position of $M_i$ Calculate the number of
      flames Evaluate the fitness function if
      $iteration == 1$ then
        F=sort(M) and OF=sort(OM)
    end
    else
        $F = sort(M_{t-1}, M_t) and OF = sort(M_{t-1}, M_t)$
    end
    for $i = 1 : n$ do
        for $j = 1 : d$ do
            Update the values of r and t Calculate the
              value of D respect to its corresponding
              moth Update $M(i, j)$ respect to
              corresponding moth
        end
    end
end
Print the best solution
```

TABLE 11. Setting the parameters for MFO algorithm.

Parameter	Value
Maximum iteration	500
Search agents NO	100

D. PRAIRIE DOG OPTIMIZATION ALGORITHM

The PDO algorithm is primarily inspired by the movements of a type of American rodent known as prairie dog (PD) [71]. The PDO algorithm mimics four prairie dog behaviors for the exploration and exploitation stages. The PDO is given exploratory behavior through the PDs' foraging and nest-building activities.

The PDs build their nests around a plentiful food resource. When a food source is exhausted, they search for a new one and construct new nests around it, thereby motivating the PDOs' exploration phase.

On the other hand, they respond to two different communications or sounds for exploitation. PDOs have sounds or signals for several situations, including the threat of predatory and abundance of food. To meet their nutritional demand and be able to defend themselves against predators, they need to be able to communicate effectively.

These two activities can lead the PDs closer to a specific location. The PDO algorithm consists of four phases: initialization, fitness function, exploration, and exploitation.

The computation complexity of the PDO algorithm is defined as follows:

$$O(t) = O((N \times dim \times T) + t_2) \tag{20}$$

1) THE REASONS FOR CHOOSING THE PDO ALGORITHM ARE

The first reason is that the PDO was tested on 22 benchmarks and ten CEC-2020. Also, it has been applied to solve twelve real-world optimization problems. The results showed that the PDO performed better than other meta-heuristic algorithms. The second reason is that we searched in ACM, Scholar, IEEE, and Science Direct databases, and this algorithm has yet to be used to estimate software effort. The third reason is that the PDO algorithm was proposed in 2022, so it was interesting to use it in optimizing parameters for the old models, such as the basic COCOMO model, which was offered in 1981, and the two Sheta's models, which were proposed in 2006.

The fourth reason is due to its advantages, which are:

• It can keep up with a balanced exploration and exploitation approach.

• It is more capable and efficient when compared to other methods.

• It can predict global optimum for real-world optimization problems with uncertain global optimal.

• It shows more consistent convergence.

• The models of the searching and burrow-building activities, which are exploration, anti-predation, and communication, which are exploitation activities, include the digging strength and predator impact features, specifically affecting the PDO updating process.

E. WHITE SHARK OPTIMIZATION ALGORITHM

The WSO is inspired by the dynamic behavior of a white shark, known as a white pointer or a huge white shark [72].

• Tracking the victim

They use their sense of smell to scan a vast area for a victim while tracking prey, as well as their sense of hearing to hear from all parts of their body.

TABLE 12. Setting the parameters for pod algorithm.

Parameter	Value
Maximum iteration	500
prairie dogs NO	100
t	1
rho	0.005
epsPD	0.1

Algorithm 4 Pseudo-Code of PDO

```
Set the parameters of PDO: n, m, p, $\epsilon$ Set G Best and
  C Best as $\phi$ Initialize the candidate solution CT and
  PD while $iter < Max - iter$ do
    for $i$=1 to $m$ do
        for $i$=1 to $m$ do
            Calculate the fitness of PD Find the best
              solution so far(C Best) Update G Best
              Update DS and PE Update $CPD_{ij}$
            if $(iter < \frac{Max_{iter}}{4})$ then
                (foraging activities)
                $PD_{i+1,j+1} = GBest_{i,j} - eGBest_{i,j} \times$
                  $p - CPD_{i,j} \times Levy(n)$
            end
            else if $(\frac{Max_{iter}}{4} < iter < \frac{Max_{iter}}{2})$ then
                (burrowing activities)
                $PD_{i+1,j+1} =$
                  $GBest_{i,j} - eGBest_{i,j} \times DS \times Levy(n)$
            end
            else if $(\frac{Max_{iter}}{2} < iter < 3\frac{Max_{iter}}{4})$ then
                (food alarm)
                $PD_{i+1,j+1} = GBest_{i,j} - eGBest_{i,j} \times$
                  $\epsilon - CPD_{i,j} \times rand$
            end
            else
                (anti predation alarm )
                $PD_{i+1,j+1} = GBest_{i,j} \times PE \times rand$
            end
        end
    end
    $iter = iter + 1$
end
Return best solution (G Best)
```

• Search for the victim (exploration)

Through two lines on both sides of the white shark, it can hear unusually, and these lines can find changes in the water pressure emitted by the victim. It can determine the electromagnetic waves produced during the prey's movement. They can pinpoint the place and size of the victim, and when they locate the victim, it moves quickly toward the target.

• Search for the victim (exploitation)

White sharks use their keen sense of smell in every possible area in the search space to identify the prey's location. When the white shark is nearer to the victim, the sense of smell increases exponentially to identify the prey's location accurately. In most cases, the white shark is tricked by the fragrance that the prey, like seals, leave behind after they depart the area to mislead the white shark. In this case, it searches randomly and explores other areas, aided by its senses.

The computation complexity of the WSO algorithm is defined as follows:

$$O(t) = O(kcn + knd) \tag{21}$$

1) THE REASONS FOR CHOOSING THE WSO ALGORITHM ARE

The first reason is that the WSO was tested on 29 benchmark test optimization problems with various dimensions related to the CEC-2017. Also, it has been applied to solve real-world optimization problems suggested for the CEC-201. The results showed that the WSO performed better than other algorithms, such as TLBO, SFS, DE, GA, GSK, AMO, PSO, BBO, and ACO [72].

The second reason is that we searched in ACM, Scholar, IEEE, and Science Direct databases, and this algorithm has yet to be used to estimate software effort. The third reason is that the WSO algorithm is new, as it was proposed in 2022, so it was interesting to use it in optimizing parameters for the old models, such as the basic COCOMO model, which was offered in 1981, and the two Sheta's models which were proposed in 2006.

The fourth reason is due to its advantages, which are:

• It's anticipated flexibility to handle various optimization problems.

• High convergence speed.

• The WSO has a small set of tuning parameters.

• The WSO is considered a strong candidate interested in creating effective, low-cost, and strong solutions to difficult optimization challenges abroad.

• The simplicity and strength of WSO are expected to make it quick and accurate to identify the solution for challenging optimization.

TABLE 13. Setting the parameters for the wso algorithm.

Parameter	Value
Maximum iteration	500
Search agents NO	100
Fmax	0.75
Fmin	0.07
tau	4.11
Pmin	0.5
Pmax	1.5
a 0	6.250
a 1	100
a 2	0.0005

Algorithm 5 Pseudo-Code of WSO

```
Initialize the parameters of the problem
Initialize the parameters of the WSO
Randomly generate the initial positions of WSO
Initialize the velocity of the initial population
Evaluate the position of the initial population
while $(iter < Max - iter)$ do
    Update the parameters v,$p_1$, $p_2$,$\mu$, a, b, $w_0$,f, mv ; and $s_s$
    for $(i=1 \ to \ n)$ do
        $v^i_{k+1} = \mu[v^i_k + p_1](w_g best_k - w^i_k) \times c_1 + p_2(w_b est^(v_k)^i - w^i_k)$
    end
    for $(i = 1 ton)$ do
        if $(rand < mv)$ then
            $w^i_{k+1} = w^i_k . \bigotimes w_0 + u.a + l.b$ else
            $w^i_{(k+1)} = w^i_k + (v^i_k)/f$
        end
        for $(i = 1 ton)$ do
            if $(rand < s_s)$ then
                $D_w = |rand < (w_{gbest_k} - w^i_k)|$ if $(i == 1)$ then
                    $w^i_{k+1} = w_g best_k + r_1 D_w sgn(r_1 - 0.5)$ else
                        $w^i_{k+1} = w_{gbest_{k+r_1}} D_w sgn(r_1 - 0.5)$
                    end
                    $w^i_{k+1} = \frac{(w^i_{k+1} + w^i_{k+1})}{(2rand)}$
                end
            end
        end
    end
    Adjust the position of the white sharks that proceed beyond the boundary.
    Evaluate and update the new positions
    $k = k + 1$.
end
Return the optimal solution obtained so far
```

VI. THE PROPOSED METHOD

The proposed models, which are used in this study, include five separate meta-heuristic algorithms. The primary objective task is to identify the generalized optimal values of all parameters for the three COCOMO-based models: basic-COCOMO, Sheta's Model 1, which is called Model I, and Sheta's Model 2, which is called Model II. One of the five meta-heuristic algorithms is applied each time, including GWO, ZOA, MFO, PDO, and WSO. The proposed method consists of two parts:

A. PART1: THE TRAINING OF MODELS

The actual effort, KLOC, and ME are the inputs. The outcomes are the optimized values for the three COCOMO-based models. The parameters of the proposed model A and B in the basic COCOMO, whereby A, B, and C are the parameters in the COCOMO Model I, and A, B, C, and D represent the parameters in the COCOMO Model II; these are set in this section, and the goal is to produce optimized coefficients using five meta-heuristic algorithms.

In other words, in the basic COCOMO, the goal is to determine an equation of independent variables [Kilo Line Of Code(KLOC)] and one dependent variable [Effort] that suits a given training sample.

The goal for the COCOMO Model I is to determine an equation of 2 independent variables [Kilo Line Of Code (KLOC), Methodology(ME)] and one dependent variable[Effort], which suits a given training sample.

Moreover, the goal of COCOMO Model II is to determine an equation of 3 independent variables [Kilo Line Of Code (KLOC), Methodology (ME), D bias factor], and one dependent variable [Effort], which suits a given training sample.

Initially, all the projects in the NASA 18 dataset were split into two groups: 70% training, which includes 13 projects, and 30% testing, which consists of 5 projects. A total of five meta-heuristic algorithms were applied separately. For the GWO Matlab implementation by [73], ZOA by [68], PDO by [71], WSO by [72], and MFO developed by [68].

Then, the meta-heuristic algorithms were used separately to optimize the coefficients to reduce the difference between each project's predicted and actual effort. Through the optimization of coefficients, in these proposed models, MAE is set as the cost function for all the meta-heuristic algorithms. The suitability function fitness is to minimize the MAE as much as possible, and meta-heuristic algorithms are repeated until the MAE is further reduced to the favored rate.

It evaluates each member in the population and gives each of them a fitness value depending on their quality. This study aims to minimize the errors, thus increasing the accuracy of the three COCOMO-based models. The reason for using the same cost function for all the meta-heuristic algorithms is to test them all under equal conditions.

For comparison, the set of parameters used in this study is discussed in [67] and [76]. The population size is unified for all the algorithms set to 100. The maximum iteration is 500. The parameters for the problem are the dimension's size, which is equal to the number of coefficients in each model, such as the dim is 2 for the basic COCOMO model, the dim is 3 for the COCOMO Model I, and the dim is 4 for COCOMO Model II. For each model, the lower bounds are -5, -30, and -30, and the upper bounds are 5, 30, and 30, respectively. All these processes are shown in Figure 1.

B. PART2: THE TESTING OF THE MODELS

This section presents the results of the training section, which are the coefficients of the three utilized COCOMO-based models used to evaluate the optimized models. The data used in this section are the testing data. The optimized coefficients are used to estimate the effort of each project. After calculating the effort for all projects, the six evaluation

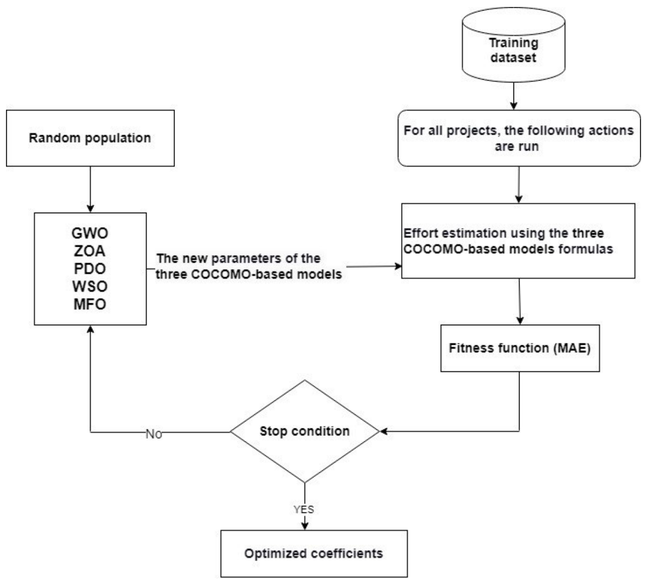
Training dataset
For all projects, the following actions are run
Random population
GWO
ZOA
PDO
WSO
MFO
The new parameters of the three COCOMO-based models
Effort estimation using the three COCOMO-based models formulas
Fitness function (MAE)
Stop condition
No
YES
Optimized coefficients

FIGURE 1. The training stage in the proposed method.

metrics are evaluated for each meta-heuristic algorithm for all the testing projects.

Once the result of each algorithm is obtained, the results are illustrated in tables and compared with the existing results in the literature. VAF, MSE, MAE, MMRE, RMSE, and R^2 of the testing dataset are compared. This phase determines the algorithm for better tuning coefficients of the three COCOMO-based models. To determine the difference between the predicted effort and the actual effort of each project, the difference should be as low as possible, which means that the accuracy of the prediction is high. All these processes are shown in Figure 2.

VII. FINDINGS

A. EXPERIMENTAL RESULTS FOR THE BASIC COCOMO ON THE NASA18 DATASET

The basic COCOMO computes software development effort, duration, and cost. In the basic COCOMO, the values A = 2.4 and B = 1.05 are obtained, respectively.

First, using the GWO on the training dataset, the following newly optimized coefficient values of A and B for the basic COCOMO model are obtained, i.e., A = 1.9045 and B = 0.9200. These new values of A and B are used to calculate the software project's effort to highlight the proposed model's effect on the software effort estimation.

Second, using the ZOA, a new optimized coefficient value of A and B for the basic COCOMO model has been found as follows: A = 3.5466 and B = 0.7876.

Third, using the PDO, a new optimized coefficient value of A and B for the basic COCOMO model is obtained, i.e., A = 3.3722 and B=0.7999.

After that, using the WSO, a new optimized coefficient value of A and B for the basic COCOMO model is obtained: A = 1.5153 and B = 0.9898.

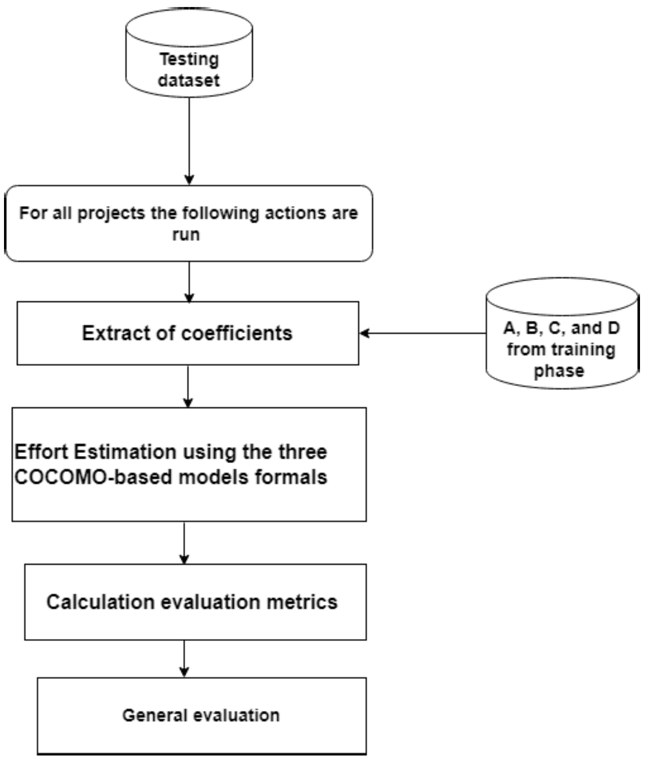
Testing dataset
For all projects the following actions are run
Extract of coefficients
A, B, C, and D from training phase
Effort Estimation using the three COCOMO-based models formals
Calculation evaluation metrics
General evaluation

FIGURE 2. The testing stage in the proposed method.

Then, using the MFO, a new optimized coefficient value of A and B for the basic COCOMO model has been found, i.e., A = 3.3339 and B = 0.8051, as shown in Table 14.

TABLE 14. BASIC COCOMO coefficient values using GWO and other algorithms.

Algorithm	A	B
GWO	1.9045	0.9200
ZOA	3.5466	0.7876
PDO	3.3722	0.7999
WSO	1.5153	0.9898
MFO	3.3339	0.8051

Table 15 illustrates the estimated effort for the NASA dataset using the new optimized coefficient values of A and B for all the five utilized algorithms. Based on Table 15, the second column indicates the actual effort provided by the NASA dataset. The third column up to the seventh column presents the estimated effort for each model, respectively. The estimated effort comparison based on Table 15 showed that all the projects have a good estimation value to the actual effort in the GWO model. It is more accurate than ZOA, PDO, WSO, and MFO.

Based on Figure 3, the suggested GWO model graph is the closest to the actual effort compared to other algorithms. This shows that the GWO model is superior to other models in estimating software's effort.

TABLE 15. Estimated effort for the basic COCOMO.

Project No.	Measured Effort	Estimate by GWO	Estimate by ZOA	Estimate by PDO	Estimate by WSO	Estimate by MFO
1	115.8	119.8315	122.9512303	123.5614606	130.5455	125.0516
2	96	64.75176	72.59124244	72.35364927	67.32266	72.97193
3	79	65.13849	72.96223945	72.72922187	67.75534	73.35318
4	90.8	75.38167	82.67954823	82.57657888	79.28369	83.3538
5	39.6	44.99047	53.15085953	52.71963068	45.50225	53.06082
6	98.4	91.77839	97.85248545	97.98811697	97.98137	99.02048
7	18.9	19.8799	26.41501015	25.9161556	18.89797	25.96374
8	10.3	16.56819	22.59960777	22.11885514	15.53359	22.13666
9	28.5	32.03497	39.74117841	39.2401671	31.57521	39.41838
10	7	5.393041	8.645869575	8.335920094	4.643532	8.289873
11	9	7.131327	10.98204834	10.62797273	6.271779	10.58597
12	7.3	12.60399	17.88240468	17.43812541	11.57428	17.42521
13	5	3.76897	6.361977462	6.10459052	3.158139	6.058587
14	8.4	8.37208	12.59856222	12.21854325	7.453138	12.18129
15	98.7	105.5771	110.3181593	110.6781315	113.9168	111.9328
16	15.6	15.4032	21.23213308	20.76022396	14.36168	20.76837
17	23.9	19.45083	25.92618024	25.42913828	18.45951	25.47269
18	138.3	132.7286	134.1954296	135.0459001	145.7216	136.7536

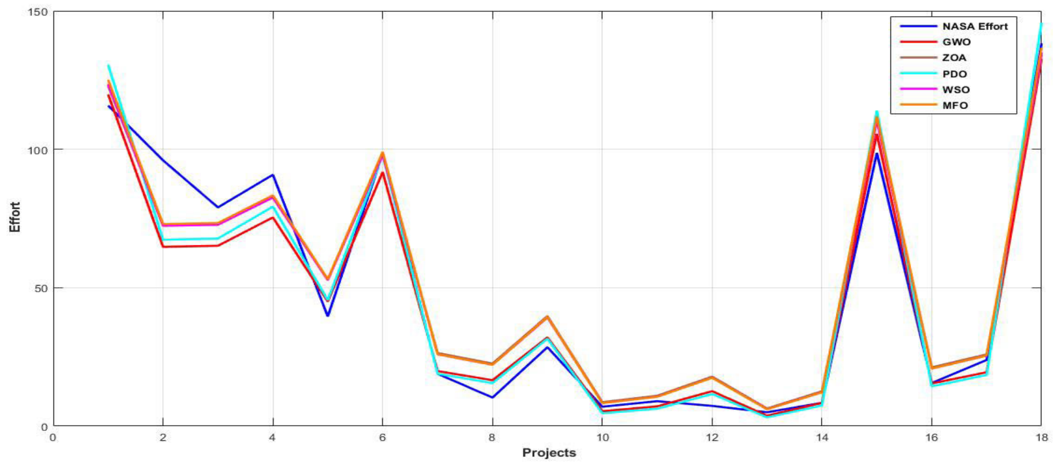

FIGURE 3. The effort graph of the basic COCOMO for the NASA, GWO, ZOA, WSO, PDO, and MFO algorithms.

The optimized coefficients are used to estimate the effort of each project, which is presented in Table 15 by replacing optimized coefficients in Equation (7) with the old ones.

After calculating the effort for all projects, the six evaluation metrics are estimated for each meta-heuristic algorithm for all the testing projects.

Once the result of each algorithm is obtained, the results are compared with the existing results in the literature, FA,

TABLE 16. Computed evaluation metrics for basic COCOMO model.

	GWO	FA	GA	PSO	ZOA	PDO	WSO	MFO
VAF	99.29%	98.16%	97.97%	97.98%	99.04%	99.09%	97.98%	99.10%
MSE	19.63	59.14	63.96	63.68	41.06	39.52	63.73	44.20
MAE	3.42	5.65	6.06	6.04	5.52	5.15	6.05	5.06
MMRE	0.06	0.11	0.13	0.12	0.22	0.20	0.13	0.20
RMSE	4.43	7.67	8.00	7.98	6.41	6.29	7.98	6.65
R^2	0.9927	0.9781	0.9763	0.9765	0.9848	0.9854	0.9764	0.9837

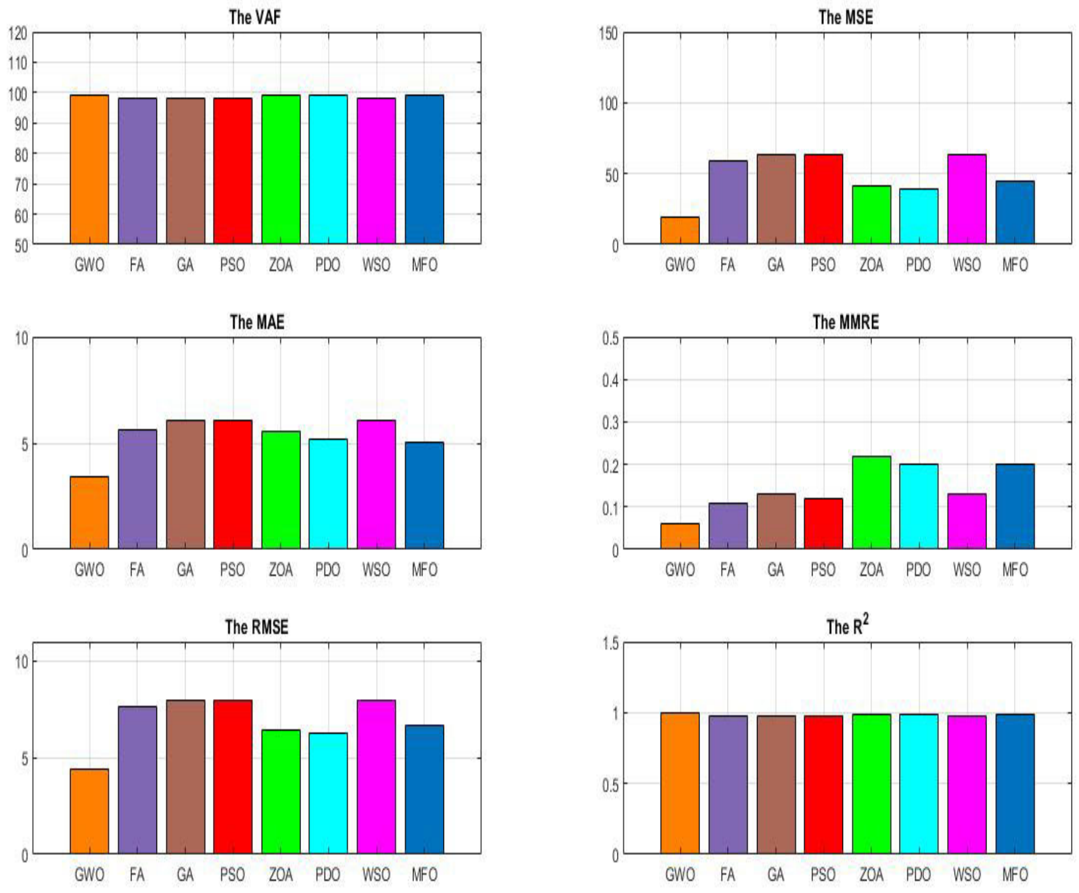

FIGURE 4. The evaluation metrics for the basic COCOMO model.

PSO, and GA. VAF, MSE, MAE, MMRE, RMSE, and R2 of the testing dataset are compared.

Here, we will explain how we obtained the results in Table 16 mathematically for each algorithm that we used, which are GWO, ZOA, PDO, WSO, and MFO:

$$\text{Act} = [8.4, 98.7, 15.6, 23.9, 138.3]$$

$$\text{Est-GWO} = [8.37208, 105.5771, 15.4032, 19.45083,$$

132.7286]

$$\text{Est-ZOA} = [12.59856222, 110.3181593, 21.23213308, 25.92618024, 134.1954296]$$

$$\text{Est-PDO} = [12.21854325, 110.6781315, 20.76022396, 25.42913828, 135.0459001]$$

$$\text{Est-WSO} = [7.453138, 113.9168, 14.36168, 18.45951, 145.7216]$$

$$\text{Est-MFO} = [12.18129, 111.9328, 20.76837, 25.47269, 136.7536]$$

A) The Variance Accounted For (VAF)

$$\begin{aligned}\text{var(act)} &= (8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2 \\ &\quad + (23.9-56.98)^2 + (138.3-56.98)^2/4 \\ &= 13520.108/4 = 3380.027.\end{aligned}$$

According to Equation (1)

1) GWO

$$\text{var(act-est)} = 23.97517.$$

$$\begin{aligned}\text{VAF-GWO} &= [1-\frac{23.97517}{3380.027}]\times 100 \\ &= [1-0.0070931889]\times 100 = 99.29\%.\end{aligned}$$

2) ZOA

$$\text{var(actest)} = 32.5601.$$

$$\text{VAFZOA} = [1-\frac{32.5601}{3380.027}]\times 100 = [1-0.0096330887] \times 100 = 99.04\%.$$

3) PDO

$$\text{var(act-est)} = 30.909692.$$

$$\text{VAF-PDO} = [1-\frac{30.909692}{3380.027}] \times 100 = [1-0.0091448062]\times 100 = 99.09\%.$$

4) WSO

$$\text{var(actest)} = 68.39592.$$

$$\text{VAFWSO} = [1-\frac{68.39592}{3380.027}] \times 100 = [1-0.0202353176]\times 100 = 97.98\%.$$

5) MFO

$$\text{var(act-est)} = 30.58404746.$$

$$\text{VAF-MFO} = [1-\frac{30.58404746}{3380.027}] \times 100 = [1-0.0090484625]\times 100 = 99.10\%.$$

B) The Mean Squared Error (MSE)

According to Equation (2)

1) MSE-GWO $= \frac{1}{5}\times(8.4-8.37208)^2+(98.7-105.5771)^2+(15.6-15.4032)^2+(23.9-19.45083)^2+(138.3-132.7286)^2 = \frac{1}{5}\times(0.02792)^2+(-6.877)^2+(0.1968)^2+(4.44917)^2+(5.5714)^2 = \frac{1}{5}\times 98.16832353 = 19.63.$

2) MSE-ZOA $= \frac{1}{5}\times(8.4-12.59856222)^2+(98.7-110.3181593)^2+(15.6-21.23213308)^2+(23.9-25.92618024)^2+(138.3-134.1954296)^2 = \frac{1}{5}\times(-4.19856)^2+(-11.6182)^2+(-5.63213)^2+(-2.02618)^2+(4.10457)^2 = \frac{1}{5}\times 205.2842659278 = 41.06.$

3) MSE-PDO $= \frac{1}{5}\times(8.4-12.21854325)^2+(98.7-110.6781315)^2+(15.6-20.76022396)^2+(23.9-25.42913828)^2+(138.3-135.0459001)^2 = \frac{1}{5}\times(-3.8185433)^2+(-11.978132)^2+(-5.160224)^2+(-1.5291383)^2+(3.2540999)^2 = \frac{1}{5}\times 14.58127+143.4756+26.62791+2.338264+10.58917 = \frac{1}{5}\times 197.612214 = 39.52.$

4) MSE-WSO $= \frac{1}{5}\times(8.4-7.453138)^2+(98.7-113.9168)^2+(15.6-14.36168)^2+(23.9-18.45951)^2+(138.3-145.7216)^2 = \frac{1}{5}\times(0.946862)^2+(-15.2168)^2+(1.23832)^2+(5.44049)^2+(-7.4216)^2 = \frac{1}{5}\times 318.660064 = 63.73.$

5) MSE-MFO $= \frac{1}{5}\times(8.4-12.18129)^2+(98.7-111.9328)^2+(15.6-20.76837)^2+(23.9-25.47269)^2+(138.3-136.7536)^2 = \frac{1}{5}\times(-3.78129)^2+(-13.2328)^2+(-5.16837)^2+(-1.57269)^2+(1.5464)^2 = \frac{1}{5}\times 220.981907 = 44.20.$

C) The Mean of Absolute Error (MAE)

According to Equation (3)

1) MAE-GWO $= \frac{1}{5}\times|8.4-8.37208|+|98.7-105.5771|+|15.6-15.4032|+|23.9-19.45083|+|138.3-132.7286| = \frac{1}{5}\times|0.02792|+|-6.877|+|0.1968|+|4.44917|+|5.5714| = \frac{1}{5}\times 17.12229 = 3.42.$

2) MAE-ZOA $= \frac{1}{5}\times|8.4-12.59856222|+|98.7-110.3181593|+|15.6-21.23213308|+|23.9-25.92618024|+|138.3-134.1954296| = \frac{1}{5}\times|-4.19856|+|-11.6182|+|-5.63213|+|-2.02618|+|4.10457| = \frac{1}{5}\times 27.57964 = 5.52.$

3) MAE-PDO $= \frac{1}{5}\times|8.4-12.21854325|+|98.7-110.6781315|+|15.6-20.76022396|+|23.9-25.42913828|+|138.3-135.0459001| = \frac{1}{5}\times|-3.8185433|+|-11.978132|+|-5.160224|+|-1.5291383|+|3.2540999| = \frac{1}{5}\times 25.7401375 = 5.15.$

4) MAE-WSO $= \frac{1}{5}\times|8.4-7.453138|+|98.7-113.9168|+|15.6-14.36168|+|23.9-18.45951|+|138.3-145.7216| = \frac{1}{5}\times|0.946862|+|-15.2168|+|1.23832|+|5.44049|+|-7.4216| = \frac{1}{5}\times 30.264072 = 6.05.$

5) MAE-MFO $= \frac{1}{5}\times|8.4-12.18129|+|98.7-111.9328|+|15.6-20.76837|+|23.9-25.47269|+|138.3-136.7536| = \frac{1}{5}\times|-3.78129|+|-13.2328|+|-5.16837|+|-1.57269|+|1.5464| = \frac{1}{5}\times 25.30155 = 5.06.$

D) The Mean Magnitude Relative Error (MMRE)

According to Equation (4)

1) MMRE-GWO $= \frac{1}{5}\times\frac{|8.4-8.37208|}{8.4}+\frac{|98.7-105.5771|}{98.7}+\frac{|15.6-15.4032|}{15.6}+\frac{|23.9-19.45083|}{23.9}+\frac{|138.3-132.7286|}{138.3} = \frac{1}{5}\times 0.312058621 = 0.06.$

2) MMRE-ZOA$= \frac{1}{5}\times\frac{|8.4-12.59856222|}{8.4}+\frac{|98.7-110.3181593|}{98.7}+\frac{|15.6-21.23213308|}{15.6}+\frac{|23.9-25.92618024|}{23.9}+\frac{|138.3-134.1954296|}{138.3}=\frac{1}{5}\times\frac{|-4.19856|}{8.4}+\frac{|-11.6182|}{98.7}+\frac{|-5.63213|}{15.6}+\frac{|-2.02618|}{23.9}+\frac{|4.10457|}{138.3}=\frac{1}{5}\times 1.0930309529=0.22.$

3) MMRE-PDO$=\frac{1}{5}\times\frac{|8.4-12.21854325|}{8.4}+\frac{|98.7-110.6781315|}{98.7}+\frac{|15.6-20.76022396|}{15.6}+\frac{|23.9-25.42913828|}{23.9}+\frac{|138.3-135.0459001|}{138.3}=\frac{1}{5}\times\frac{|-3.8185433|}{8.4}+\frac{|-11.978132|}{98.7}+\frac{|-5.160224|}{15.6}+\frac{|-1.5291383|}{23.9}+\frac{|3.2540999|}{138.3}=\frac{1}{5}\times 0.9942410301=0.20.$

4) MMRE-WSO $=\frac{1}{5}\times\frac{|8.4-7.453138|}{8.4}+\frac{|98.7-111.9328|}{98.7}+\frac{|15.6-14.36168|}{15.6}+\frac{|23.9-18.45951|}{23.9}+\frac{|138.3-145.7216|}{138.3}=\frac{1}{5}\times\frac{|0.946862|}{8.4}+\frac{|-15.2168|}{98.7}+\frac{|1.23832|}{15.6}+\frac{|5.44049|}{23.9}+\frac{|-7.4216|}{138.3}==\frac{1}{5}\times 0.6275720091=0.13.$

5) MMRE-MFO $=\frac{1}{5}\times\frac{|8.4-12.18129|}{8.4}+\frac{|98.7-111.9328|}{98.7}+\frac{|15.6-20.76837|}{15.6}+\frac{|23.9-25.47269|}{23.9}+\frac{|138.3-136.7536|}{138.3}=\frac{1}{5}\times\frac{|-3.78129|}{8.4}+\frac{|-13.2328|}{98.7}+\frac{|-5.16837|}{15.6}+|\frac{-1.57269|}{23.9}+\frac{|1.5464|}{138.3}=\frac{1}{5}\times 0.992514681=0.20.$

E) The Root Mean Squared Error (RMSE)

According to Equation (5)

1) $RMSE-GWO=\sqrt{19.63}=4.43.$
2) $RMSE-ZOA=\sqrt{41.06}=6.41.$
3) $RMSE-PDO=\sqrt{39.52}=6.29.$
4) $RMSE-WSO=\sqrt{63.733}=7.98.$
5) $RMSE-MFO=\sqrt{44.20}=6.65.$

F) The R-Squared (R^2)

According to Equation (6)

$$Mean(act)=\frac{(8.4+98.7+15.6+23.9+138.3)}{5}=56.98.$$

1) $R^2-GWO=((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)-98.16832353)/((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)=\frac{(13520.108-98.16832353)}{13520.108}=0.9927.$

2) $R^2-ZOA=(8.4-56.98+98.7-56.98+15.6-56.98+23.9-56.98+138.3-56.98)^2-205.283346)/(8.4-56.98+98.7-56.98+15.6-56.98+23.9-138.3)^2=\frac{(13520.108-205.283346)}{13520.108}=0.9848.$

3) $R^2-PDO=(8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)-197.612071)/(8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)=\frac{(13520.108-197.612071)}{13520.108}=0.9854.$

4) $R^2-WSO=(8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)-318.660064)/(8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)=\frac{(13520.108-318.660064)}{13520.108}=0.9764.$

5) $R^2-MFO=(8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)-220.981907/(8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)=\frac{(13520.108-220.981907)}{13520.108}=0.9837.$

Table 16 compares the GWO model's accuracy with ZOA, PDO, WSO, MFO, Firefly, PSO, and GA models found in previous studies [76]. Using six different estimation metrics. From the evaluation metrics, the VAF value is expected to be high because it is associated with the fit quality of the estimation.

Moreover, if $R^2 = 1$, this indicates that 100% percent of the increase in the dependent variable is because of an increase in the independent variable, and if $R^2 = 0$, there is no relationship between the variables. Regarding the metric R^2, the closer the value is to one, the better the data fit the model can provide. For the rest of the evaluation metrics, the value should be as low as possible to be better because it is a relative error found in the estimation process.

Based on Table 16, it can be observed that the VAF of the GWO model is 99.29%, which is the maximum value found when compared with other models. The VAF of Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 98.16%, 97.97%, 97.98%, 99.04%, 99.09%, 97.98%, and 99.10%, respectively.

Also, the R^2 of the GWO model is 0.9927, and the R^2 of Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 0.9781, 0.9763, 0.9765, 0.9848, 0.9854, 0.9764, and 0.9837, respectively. According to the comparison, the GWO model has lower relative errors and higher VAF and R^2 values.

For example, the MSE of GWO is 19.63, and the MSE for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 59.14, 63.96, 63.68, 41.06, 39.52, 63.73, 44.20, respectively. These values show that the GWO model can reduce errors by 39.51, 44.33, 19.63, 21.43, 19.63, 44.1, and 24.57, respectively.

By using the MAE as an evaluation metric, the MAE of the GWO model is 3.42, which obtained better results than other models. The MAE for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 5.65, 6.06, 6.04, 5.52, 5.15, 6.05, and 5.06, respectively. Compared with the GWO model, the GWO model could reduce 2.23, 2.64, 2.62, 2.1, 1.73, 2.63, and 1.64 percent of mean absolute errors.

Regarding MMRE, the GWO model obtained better results by 0.05, 0.07, 0.06, 0.16, 0.14, 0.07, and 0.14.

By using the RMSE as an evaluation metric, the RMSE of the GWO model is 4.43, and the RMSE for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 7.67, 8.00, 7.98, 6.41, 6.29, 7.98, and 6.65, respectively. Compared with the GWO model, the GWO model could reduce errors by 3.24, 3.57, 3.55, 1.98, 1.86, 3.55, and 2.22, respectively.

The GWO model has lower relative errors, higher VAF, and R^2 values in the six evaluation metrics. Therefore, the GWO model is efficient, and the basic COCOMO model software projects' effort should be estimated with the new parameters.

B. EXPERIMENTAL RESULTS FOR THE COCOMO MODEL I ON THE NASA18 DATASET

COCOMO Model I is a new model structure that can be used to estimate the software effort for projects proposed by [67]. It considered the effect of the methodology as linearly related to effort estimation. It has three parameters: A, B, and C.

First, using the GWO, the following newly optimized coefficient values of A, B, and C for the COCOMO Model I are obtained, i.e., A = 1.9496, B = 0.9165, and C = 0.0287. These new values of A, B, and C are used to calculate the software project's effort to highlight the effect of the proposed models on the software effort estimation.

Second, using the ZOA, a new optimized coefficient value of A, B, and C for the COCOMO Model I has been found: A = 8.7353, B = 0.6133, and C = −0.6982.

Third, using the PDO, a new optimized coefficient value of A, B, and C for the COCOMO Model I model is obtained, i.e., A = 14.8136, B = 0.5151, and C = −1.1591.

After that, using the WSO, a new optimized coefficient value of A, B, and C for the COCOMO Model I is obtained: A = 8.9746, B = 0.6111, and C = −0.7792.

Then, using the MFO, a new optimized coefficient value of A, B, and C for the COCOMO Model I have been found, i.e.,. A = 12.8178, B = 0.541, and C = −1.0211, as shown in Table 17.

TABLE 17. Cocomo modeli coefficient values using GWO and other algorithms.

Algorithm	A	B	C
GWO	1.9496	0.9165	0.0287
ZOA	8.7353	0.6133	-0.6982
PDO	14.8136	0.5151	-1.1591
WSO	8.9746	0.6111	-0.7792
MFO	12.8178	0.541	-1.0211

Table 18 illustrates the estimated effort for the NASA dataset using the new optimized coefficient parameters' values of A, B, and C for all the five applied algorithms. The estimated effort comparison based on Table 18 showed that all the projects have a good estimation value to the actual effort in the GWO model, and it is more accurate than ZOA, PDO, WSO, and MFO.

As shown in Figure 5, the proposed GWO model graph is almost in line with the actual effort. This demonstrates that the GWO model is superior to other models in estimating the effort of software.

The optimized coefficients are used to estimate the effort of each project, which is presented in Table 18 by replacing the optimized coefficients in Equation (8) with the old ones. After the calculation of the effort for all projects,the six evaluation metrics are estimated for each meta-heuristic algorithm for all the testing projects.

Once the result of each algorithm is obtained, the results are compared with the existing results in the literature, FA, PSO, and GA. The testing dataset's VAF, MSE, MAE, MMRE, RMSE, and R2 are compared. Here, we will explain how we obtained the results in Table 19 mathematically for each algorithm that we used, which are GWO, ZOA, PDO, WSO, and MFO:

$$\begin{aligned}\text{Act} &= [8.4, 98.7, 15.6, 23.9, 138.3]\\ \text{Est-GWO} &= [9.354496, 107.4434, 16.41796,\\ &\quad 20.5111, 134.6714]\\ \text{Est-ZOA} &= [3.19212, 102.5445, 16.3423,\\ &\quad 22.26478, 124.1721]\\ \text{Est-PDO} &= [0.32518, 99.71053, 16.45137,\\ &\quad 23.11433, 120.0474]\\ \text{Est-WSO} &= [1.400129, 101.9415, 14.93913,\\ &\quad 20.97006, 123.9356]\\ \text{Est-MFO} &= [1.004651, 100.167, 16.24875,\\ &\quad 22.69254, 120.7666]\end{aligned}$$

A) The Variance Accounted For (VAF)

$$\begin{aligned}\text{var(act)} &= (8.4-56.98)^2 + (98.7-56.98)^2\\ &\quad + (15.6-56.98)^2 + (23.9-56.98)^2\\ &\quad + (138.3-56.98)^2/4\\ &= 13520.108/4 = 3380.027.\end{aligned}$$

According to Equation (1)

1) GWO

var (act-est) = 25.07571.

$$\text{VAF-GWO} = [1 - \frac{25.05771}{3380.027}] \times 100 = [1-0.0074134644] \times 100 = 99.26\%.$$

2) ZOA

var(act-est) = 47.75908.

$$\text{VAF-ZOA} = [1 - \frac{47.75908}{3380.027}] \times 100 = [1-0.0141297925] \times 100 = 98.59\%.$$

3) PDO

var(act-est) = 68.29972.

$$\text{VAF-PDO} = [1 - \frac{68.29972}{3380.027}] \times 100 = [1-0.0202068563] \times 100 = 97.98\%.$$

4) WSO

var(act-est) = 45.14172.

$$\text{VAF-WSO} = [1 - \frac{45.14172}{3380.027}] \times 100 = [1-0.0133554318] \times 100 = 98.66\%.$$

5) MFO

var(act-est) = 62.68643.

$$\text{VAF-MFO} = [1 - \frac{62.68643}{3380.027}] \times 100 = [1 - 0.0185461329] \times 100 = 98.15\%.$$

TABLE 18. Estimated effort for the cocomo model I.

Project No.	Measured Effort	Estimate by GWO	Estimate by ZOA	Estimate by PDO	Estimate by WSO	Estimate by MFO
1	115.8	121.6124	117.2215	115.814	117.1775	115.7802
2	96	65.97583	77.70083	83.50652	77.80115	81.52743
3	79	66.33625	78.76363	85.02191	78.95045	82.90614
4	90.8	76.66844	87.47613	92.98384	87.72235	91.05981
5	39.6	46.50964	47.47275	46.44397	46.05113	46.56094
6	98.4	93.40916	95.41397	96.08461	95.1374	95.54847
7	18.9	20.91609	23.5654	24.9422	22.36253	24.36269
8	10.3	17.79733	13.20769	10.3271	11.26997	11.02057
9	28.5	33.33303	35.69641	36.01274	34.35994	35.74633
10	7	6.245134	-0.66964	-3.60515	-2.34134	-2.909
11	9	7.808926	7.797025	9.000995	6.766819	8.459757
12	7.3	13.69973	9.145074	6.743393	7.334905	7.289552
13	5	4.651817	-5.78084	-10.746	-7.69472	-9.44233
14	8.4	9.354496	3.19212	0.32518	1.400129	1.004651
15	98.7	107.4434	102.5445	99.71053	101.9415	100.167
16	15.6	16.41796	16.3423	16.45137	14.93913	16.24875
17	23.9	20.5111	22.26478	23.11433	20.97006	22.69254
18	138.3	134.6714	124.1721	120.0474	123.9356	120.7666

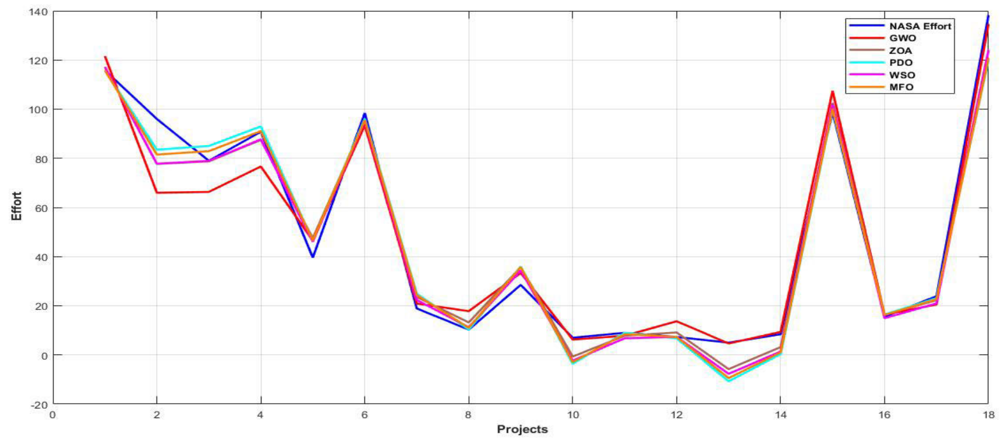

FIGURE 5. The effort graph of COCOMO Model I for the NASA, GWO, ZOA, WSO, PDO, and MFO.

B) The Mean Squared Error (MSE)

According to Equation (2)

1) MSE-GWO=$\frac{1}{5} \times (8.4 - 9.354496)^2 + (98.7 - 107.4434)^2 + (15.6 - 16.41796)^2 + (23.9 - 20.5111)^2 + (138.3 - 134.6714)^2 = \frac{1}{5} \times (-0.954496)^2 + (-8.7434)^2 + (-0.81796)^2 + (3.3889)^2 + (3.6286)^2 = \frac{1}{5} \times 102.6785459 = 20.54$.

2) MSE-ZOA=$\frac{1}{5} \times (8.4 - 3.19212)^2 + (98.7 - 102.5445)^2 + (15.6 - 16.3423)^2 + (23.9 - 22.26478)^2 + (138.3 - 124.1721)^2 = \frac{1}{5} \times (5.20788)^2 + (-3.8445)^2 + (-0.7423)^2 + (1.63522)^2 + (14.1279)^2 = \frac{1}{5} \times 244.7247065 = 48.95$.

3) MSE-PDO=$\frac{1}{5} \times (8.4 - 0.32518)^2 + (98.7 - 99.71053)^2 + (15.6 - 16.45137)^2 + (23.9 - 23.11433)^2 + (138.3 -$

TABLE 19. Computed evaluation metrics for the COCOMO model I.

	GWO	FA	GA	PSO	ZOA	PDO	WSO	MFO
VAF	99.26%	98.62%	97.97%	98.528%	98.59%	97.98 %	98.66%	98.15%
MSE	20.54	47.74	98.17	60.07	48.95	80.14	54.97	73.23
MAE	3.51	5.56	7.70	5.63	5.11	5.8	5.64	5.65
MMRE	0.08	0.24	0.29	0.23	0.18	0.24	0.23	0.22
RMSE	4.53	6.82	9.39	7.72	7	8.95	7.41	8.56
R^2	0.9924	0.9823	0.9637	0.9778	0.9819	0.9704	0.9797	0.9729

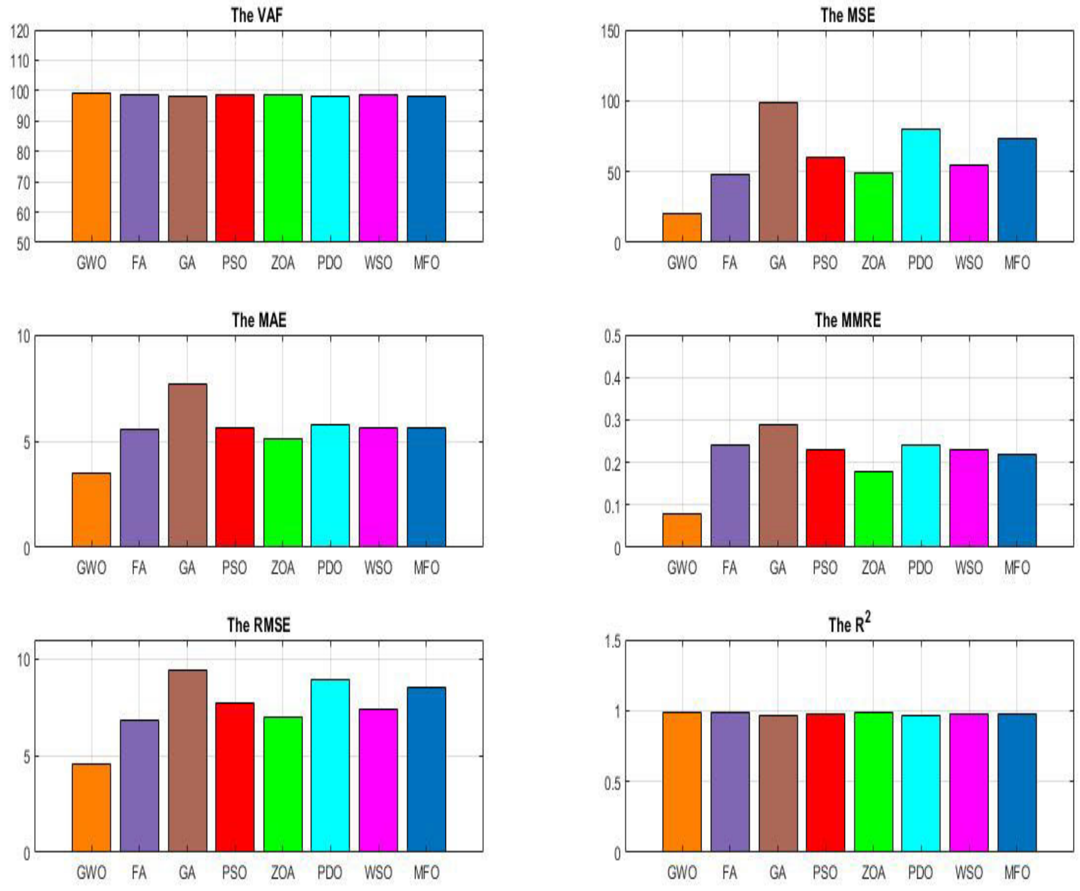

FIGURE 6. The evaluation metrics for the COCOMO Model I.

$120.0474)^2 = \frac{1}{5} \times (8.07482)^2 + (-1.01053)^2 + (-0.85137)^2 + (0.78567)^2 + (18.2526)^2 = \frac{1}{5} \times 400.7234038991 = 80.14$.

4) MSE-WSO=$\frac{1}{5} \times (8.4 - 1.400129)^2 + (98.7 - 101.9415)^2 + (15.6 - 14.93913)^2 + (23.9 - 20.97006)^2 + (138.3 - 123.9356)^2 = \frac{1}{5} \times 274.8628011871 = 54.97$.

5) MSE-MFO=$\frac{1}{5}\times(8.4-1.004651)^2+(98.7-100.167)^2+(15.6-16.24875)^2+(23.9-22.69254)^2+(138.3-120.7666)^2=\frac{1}{5}\times 366.1422276059=73.23$.

C) The Mean of Absolute Error (MAE)

According to Equation (3)

1) MAE-GWO=$\frac{1}{5}\times|8.4-9.354496|+|98.7-107.4434|+|15.6-16.41796|+|23.9-20.5111|+|138.3-134.6714|=\frac{1}{5}\times 17.533356=3.51$.

2)MAE-ZOA=$\frac{1}{5}\times|8.4-3.19212|+|98.7-102.5445|+|15.6-16.3423|+|23.9-22.26478|+|138.3-124.1721|=\frac{1}{5}\times|5.20788|+|-3.8445|+|-0.7423|+|1.63522|+|14.1279|=\frac{1}{5}\times 25.5578=5.11$.

3) MAE-PDO=$\frac{1}{5}\times|8.4-0.32518|+|98.7-99.71053|+|15.6-16.45137|+|23.9-23.11433|+|138.3-120.0474|=\frac{1}{5}\times|8.07482|+|-1.01053|+|-0.85137|+|0.78567|+|18.2526|=\frac{1}{5}\times 28.97499=5.8$.

4) MAE-WSO=$\frac{1}{5}\times|8.4-1.400129|+|98.7-101.9415|+|15.6-14.93913|+|23.9-20.97006|+|138.3-123.9356|=\frac{1}{5}\times 28.19658=5.64$.

5) MAE-MFO=$\frac{1}{5}\times|8.4-1.004651|+|98.7-100.167|+|15.6-16.24875|+|23.9-22.69254|+|138.3-120.7666|=\frac{1}{5}\times 28.251959=5.65$.

D) The Mean Magnitude Relative Error (MMRE)

According to Equation (4)

1) MMRE-GWO=$\frac{1}{5}\times\frac{|8.4-9.354496|}{8.4}+\frac{|98.7-107.4434|}{98.7}+\frac{|15.6-16.41796|}{15.6}+\frac{|23.9-20.5111|}{23.9}+\frac{|138.3-134.6714|}{138.3}=\frac{1}{5}\times 0.4226815672=0.08$

2) MMRE-ZOA=$\frac{1}{5}\times\frac{|8.4-3.19212|}{8.4}+\frac{|98.7-102.5445|}{98.7}+\frac{|15.6-16.3423|}{15.6}+\frac{|23.9-22.26478|}{23.9}+\frac{|138.3-124.1721|}{138.3}=\frac{1}{5}\times 0.8770936753=0.18$.

3) MMRE-PDO=$\frac{1}{5}\times\frac{|8.4-0.32518|}{8.4}+\frac{|98.7-99.71053|}{98.7}+\frac{|15.6-16.45137|}{15.6}+\frac{|23.9-23.11433|}{23.9}+\frac{|138.3-120.0474|}{138.3}=\frac{1}{5}\times 1.1909530242=0.24$.

4) MMRE-WSO=$\frac{1}{5}\times\frac{|8.4-1.400129|}{8.4}+\frac{|98.7-101.9415|}{98.7}+\frac{|15.6-14.93913|}{15.6}+\frac{|23.9-20.97006|}{23.9}+\frac{|138.3-123.9356|}{138.3}=\frac{1}{5}\times 1.1349790784=0.23$.

5) MMRE-MFO=$\frac{1}{5}\times\frac{|8.4-1.004651|}{8.4}+\frac{|98.7-100.167|}{98.7}+\frac{|15.6-16.24875|}{15.6}+\frac{|23.9-22.69254|}{23.9}+\frac{|138.3-120.7666|}{138.3}=\frac{1}{5}\times 1.1141478085=0.22$.

E) The Root Mean Squared Error (RMSE)

According to Eq (5)

1)$RMSE-GWO=\sqrt{20.54}=4.53$.

2)$RMSE-ZOA=\sqrt{48.95}=7$.

3)$RMSE-PDO=\sqrt{80.14}=8.95$.

4)$RMSE-WSO=\sqrt{54.97}=7.41$.

5)$RMSE-MFO=\sqrt{73.23}=8.56$.

F)The R-Squared (R^2)

According to Equation (6)

$Mean(act)=\frac{(8.4+98.7+15.6+23.9+138.3)}{5}=56.98$.

1) $R^2-GWO=((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)-102.6785459)/((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)=\frac{(13520.108-102.6785459)}{13520.108}=0.9924$.

2) $R^2-ZOA=((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)-244.7247065)/((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)=\frac{(13520.108-244.7247065)}{13520.108}=0.9819$

3) $R^2-PDO=(((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)-400.7234030)/((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)=\frac{(13520.108-400.7234030)}{13520.108}=0.9704$.

4) $R^2-WSO=((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)-273.1948372)/((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)=\frac{(13520.108-273.1948372)}{13520.108}=0.9797$.

5) $R^2-MFO=((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)-367.6289426/((8.4-56.98)^2+(98.7-56.98)^2+(15.6-56.98)^2+(23.9-56.98)^2+(138.3-56.98)^2)=\frac{13520.108-367.6289426)}{13520.108}=0.9729$.

Table 19. presents the accuracy comparison of the GWO model with ZOA, PDO, WSO, MFO, Firefly, PSO, and GA models established in previous studies [76].

Using six different estimation metrics for all the testing projects. First, using the VAF as an evaluation metric, the VAF of GWO is 99.26%, which is the maximum value when compared with ZOA, PDO, WSO, MFO, PSO, and GA models.

From the evaluation metrics, the VAF value for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models is 98.62%, 97.97%, 98.528%, 98.59%, 97.98%, 98.66%, and 98.15%, respectively. The model generated by the GWO algorithm has the best quality among all other models.

Second, by using the MSE as an evaluation metric, the MSE of GWO is 20.54, and the MSE for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 47.74, 98.17, 60.07, 48.95, 80.14, 54.97, and 73.23, respectively. These values indicate that the GWO model can reduce 27.2, 77.63, 39.53, 28.41, 59.6, 34.43, and 52.69 errors, respectively.

Third, by using the MAE as an evaluation metric, the MAE of GWO is 3.51, and the MAE for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 5.56, 7.70, 5.63, 5.11, 5.8, 5.64, and 5.65, respectively. These values indicate that the GWO model can reduce errors by 2.05, 4.19, 2.12, 1.6, 2.29, 2.13, and 2.14, respectively.

Fourth, the MMRE of GWO is 0.08, and for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 0.24, 0.29, 0.23, 0.18, 0.24, 0.23, 0.22, respectively. This means the GWO can reduce the errors by 0.16, 0.21, 0.15, 0.1, 0.16, 0.15, and 0.14, respectively.

Fifth, the RMSE of GWO is 4.53, and for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 6.82, 9.39, 7.72, 7, 8.95, 7.41, and 8.56, respectively. This means the GWO can reduce the errors by 2.29, 4.86, 3.19, 2.47, 4.42, 2.88, and 4.03, respectively.

Sixth, the R^2 as an evaluation metric is used to measure the extent of the relationship between the independent variable and the dependent variable. It can be observed that GWO has the highest value among all other models. The R^2 of the GWO is 0.9924%, while the Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 0.9823, 0.9637, 0.9778, 0.9819, 0.9704, 0.9797, and 0.9729, respectively.

Based on the results of previous evaluation metrics, it can be concluded that the GWO algorithm obtained the highest value in the VAF and R2 and the lowest value in relative errors. Therefore, the COCOMO Model I, proposed by [67] as an extension of the basic COCOMO software projects effort, should be estimated with the new parameter values generated by the GWO model.

C. EXPERIMENTAL RESULTS FOR THE COCOMO MODEL II ON THE NASA18 DATASET

The COCOMO Model II is a new model structure that can be used to estimate project software effort by adding a new bias called D, which was proposed by [67]. It is a modified version of the famous COCOMO model. It is an extension of the famous basic COCOMO by adding a new bias parameter to the basic COCOMO. It has four parameters: A, B, C, and D.

First, using the GWO, the following newly optimized coefficient values of A, B, C, and D for COCOMO Model II are obtained. A = 2.5663, B = 0.8736, C=-1.2078, and D=29.5778. These new values of A, B, C, and D are used to calculate the software projects' effort to show the proposed models' effect on software effort estimation.

Second, using the ZOA, a new optimized coefficient value of A, B, C, and D for the COCOMO Model II has been found. A=4.1863, B=0.7751, C=−0.8405, and D=12.783.

Third, using the PDO, a new optimized coefficient value of A, B, C, and D for the COCOMO Model II is found. A = 7.8707, B = 0.63641, C=-0.89707, and D=5.7518.

TABLE 20. Cocomo model II coefficient values using GWO and other algorithms.

Algorithm	A	B	C	D
GWO	2.5663	0.8736	-1.2078	29.5778
ZOA	4.1863	0.7751	-0.8405	12.783
PDO	7.8707	0.63641	-0.89707	5.7518
WSO	2.1805	0.91133	-1.1185	27.803
MFO	2.93417	0.841854	-1.18803	27.5938

After that, using the WSO, a new optimized coefficient value of A, B, C, and D for COCOMO Model II has been found. A=2.1805, B=0.91133, C=-1.1185, and D=27.803. Then, using the MFO, a new optimized coefficient value of A, B, C, and D for the COCOMO Model II is found. A=2.93417, B=0.841854, C=-1.18803, and D=27.5938, as shown in Table 20.

Table 21. displays the estimated effort for the NASA dataset using the new optimized coefficient values of A, B, and C for all the five applied algorithms. The estimated effort comparison based on Table 21 shows that all the projects have a good estimation value to the actual effort in the GWO model, which is more accurate than other models.

As shown in Figure 7, the proposed GWO model graph almost aligns with the actual effort. This indicates that the GWO model is more efficient in estimating the effort of software than other models. The optimized coefficients are used to estimate the effort of each project, which is presented in Table 21 by replacing optimized coefficients in Equation (9) with the old ones.

After calculating the effort for all projects, the six evaluation metrics are estimated for each meta-heuristic algorithm for all the testing projects. Once the result of each algorithm is obtained, the results are compared with the existing results in the literature, FA, PSO, and GA. The testing dataset's VAF, MSE, MAE, MMRE, RMSE, and R2 are compared.

Here, we will explain how we obtained the results in Table 22 mathematically for each algorithm that we used, which are GWO, ZOA, PDO, WSO, and MFO:

$$\begin{aligned}\text{Act} &= [8.4, 98.7, 15.6, 23.9, 138.3]\\ \text{Est-GWO} &= [5.02114, 103.4894, 15.6461, 20.27858,\\ &\quad 132.9004]\\ \text{Est-ZOA} &= [2.98328, 106.6681, 14.44949, 19.74091,\\ &\quad 133.7306]\\ \text{Est-PDO} &= [1.656988, 100.9093, 14.95078, 20.80425,\\ &\quad 123.5154]\\ \text{Est-WSO} &= [4.819038, 105.0444, 14.89492, 19.39078,\\ &\quad 135.7797]\\ \text{Est-MFO} &= [4.515002, 101.6651, 15.38723, 20.11632,\\ &\quad 129.7965]\end{aligned}$$

A) The Variance Accounted For (VAF)

$$\begin{aligned}\text{var(act)} &= (8.4-56.98)^2\\ &\quad +(98.7-56.98)^2+(15.6-56.98)^2\\ &\quad +(23.9-56.98)^2+(138.3-56.98)^2/4\\ &= 13520.108/4 = 3380.027.\end{aligned}$$

According to Eq (1)

1) GWO

$$\text{var(act-est)} = 16.29589.$$

$$\text{VAF-GWO} = [1-\frac{16.29589}{3380.027}] \times 100 = [1-0.0048212307]\times 100 = 99.52\%.$$

2) ZOA

$$\text{var(act-est)} = 30.39845.$$

$$\text{VAF-ZOA} = [1-\frac{30.39845}{3380.027}] \times 100 = [1-0.0089935524] \times 100 = 99.101\%.$$

TABLE 21. Estimated effort for the COCOMO model II.

Project No.	Measured Effort	Estimate by GWO	Estimate by ZOA	Estimate by PDO	Estimate by WSO	Estimate by MFO
1	115.8	124.3753	124.7544	116.9819	126.1935	121.8154
2	96	78.45808	77.64898	78.05229	77.1454	77.77172
3	79	80.08003	78.90026	79.32184	78.68815	79.36374
4	90.8	89.79867	88.8079	88.05788	88.79849	88.80555
5	39.6	38.99199	43.46473	44.50327	38.65359	39.00002
6	98.4	96.26711	97.98658	94.60517	96.67724	94.88099
7	18.9	21.9744	21.13152	22.29859	20.98531	21.80044
8	10.3	8.530494	10.10925	10.40005	8.360454	8.441908
9	28.5	29.57512	31.87185	33.40399	28.84422	29.59814
10	7	5.070424	0.992011	-1.40162	4.836328	4.310736
11	9	15.61998	9.545918	8.325502	14.61534	14.84256
12	7.3	7.575786	7.300275	7.033137	7.305336	7.303468
13	5	0.666198	-3.31084	-6.74565	0.772501	-0.19147
14	8.4	5.02114	2.98328	1.656988	4.819038	4.515002
15	98.7	103.4894	106.6681	100.9093	105.0444	101.6651
16	15.6	15.6461	14.44949	14.95078	14.89492	15.38723
17	23.9	20.27858	19.74091	20.80425	19.39078	20.11632
18	138.3	132.9004	133.7306	123.5154	135.7797	129.7965

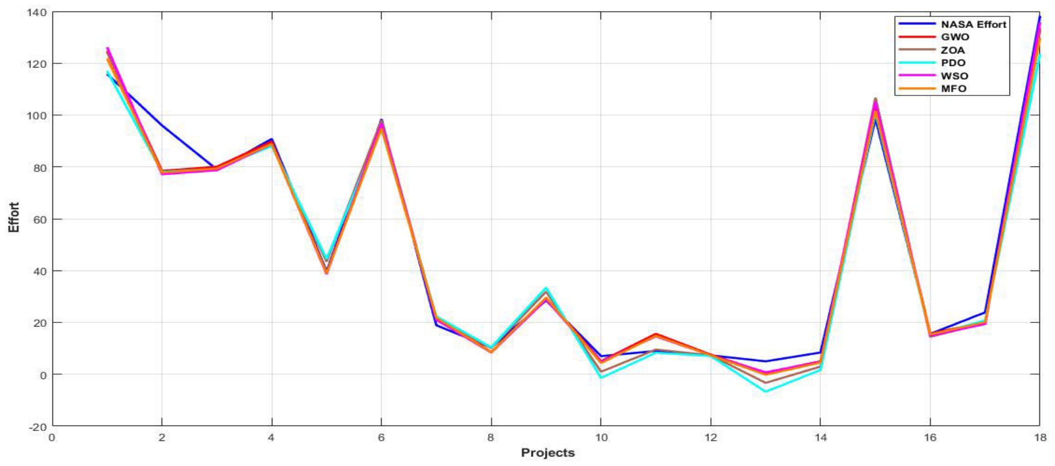

FIGURE 7. The effort graph of COCOMO Model II for the NASA, GWO, ZOA, WSO, PDO, and MFO.

3) PDO

var(act-est) $= 43.13894$.

$$\text{VAF-PDO} = [1 - \frac{43.13894}{3380.027}] \times 100 = [1 - 0.012762898 \times 100 = 98.72\%.$$

TABLE 22. Computed evaluation metrics for the COCOMO model II.

	GWO	FA	GA	PSO	ZOA	PDO	WSO	MFO
VAF	99.52%	98.63%	97.60%	98.70%	99.101%	98.72%	99.44%	99.45%
MSE	15.33	45.02	114.79	52.85	26.47	55.79	16.05	22.11
MAE	3.45	5.57	7.83	5.29	4.65	5.5	3.53	3.87
MMRE	0.13	0.24	0.27	0.21	0.20	0.22	0.15	0.15
RMSE	3.91	6.62	9.86	7.19	5.14	7.47	4.01	4.70
R^2	0.9943	0.9833	0.9575	0.9805	0.9902	0.9794	0.9941	0.9918

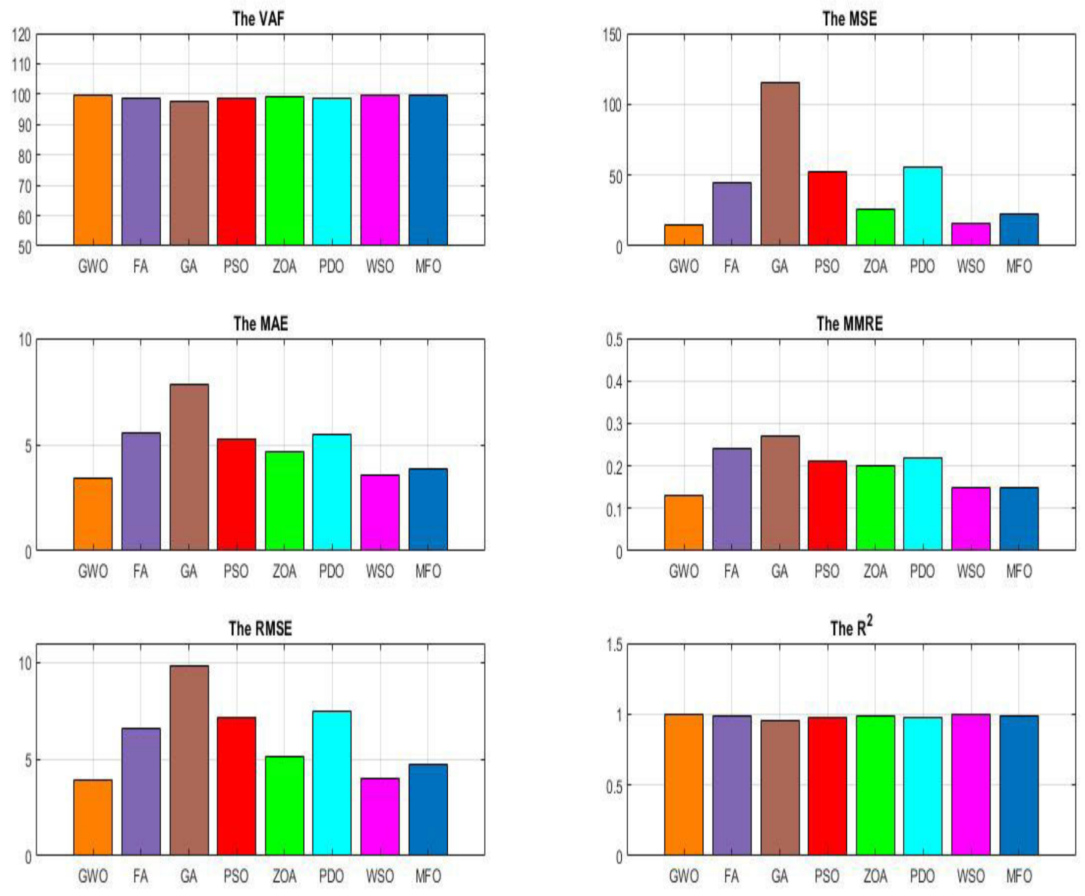

FIGURE 8. The evaluation metrics for the COCOMO Model II.

4) WSO

var(act-est) = 18.82858

$$\text{VAF-WSO} = [1 - \frac{18.82858}{3380.027}] \times 100 = [1 - 0.0055705413] \times 100 = 99.44\%.$$

5) MFO

var(act-est) = 18.6344.

$$\text{VAF-MFO} = [1 - \frac{18.6344}{3380.027}] \times 100 = [1 - 0.0055130921] \times 100 = 99.45\%.$$

B) The Mean Squared Error (MSE)

According to Equation (2)

1)MSE-GWO $= \frac{1}{5} \times (8.4 - 5.02114)^2 + (98.7 - 103.4894)^2 + (15.6 - 15.6461)^2 + (23.9 - 20.27858)^2 + (138.3 - 132.9004)^2 = \frac{1}{5} \times (3.37886)^2 + (-4.7894)^2 + (-0.0461)^2 + (3.62142)^2 + (5.3996)^2 = \frac{1}{5} \times 76.62753545 = 15.33$.

2)MSE-ZOA $= \frac{1}{5} \times (8.4 - 2.98328)^2 + (98.7 - 106.6681)^2 + (15.6 - 14.44949)^2 + (23.9 - 19.74091)^2 + (138.3 - 133.7306)^2 = \frac{1}{5} \times (5.41672)^2 + (-7.9681)^2 + (1.15051)^2 + (4.15909)^2 + (4.5694)^2 = \frac{1}{5} \times 17.62792 + 134.9816 + 31.72092 + 4.105406 + 16.8475 = \frac{1}{5} \times 132.3325924 = 26.47$.

3)MSE-PDO $= \frac{1}{5} \times (8.4 - 1.656988)^2 + (98.7 - 100.9093)^2 + (15.6 - 14.95078)^2 + (23.9 - 20.80425)^2 + (138.3 - 123.5154)^2 = \frac{1}{5} \times (6.743012)^2 + (-2.2093)^2 + (0.64922)^2 + (3.09575)^2 + (14.7846)^2 = \frac{1}{5} \times 278.9387692 = 55.79$.

4)MSE-WSO $= \frac{1}{5} \times (8.4 - 4.819038)^2 + (98.7 - 105.0444)^2 + (15.6 - 14.89492)^2 + (23.9 - 19.39078)^2 + (138.3 - 135.7797)^2 = \frac{1}{5} \times (3.580962)^2 + (-6.3444)^2 + (0.70508)^2 + (4.50922)^2 + (2.5203)^2 = \frac{1}{5} \times 80.25681511 = 16.05$.

5)MSE-MFO $= \frac{1}{5} \times (8.4 - 4.515002)^2 + (98.7 - 101.6651)^2 + (15.6 - 15.38723)^2 + (23.9 - 20.11632)^2 + (138.3 - 129.7965)^2 = \frac{1}{5} \times (3.884998)^2 + (-2.9651)^2 + (0.21277)^2 + (3.78368)^2 + (8.5035)^2 = \frac{1}{5} \times 110.5560451 = 22.11$.

C) The Mean of Absolute Error (MAE)

According to Equation (3)

1) MAE-GWO $= \frac{1}{5} \times |8.4 - 5.02114| + |98.7 - 103.4894| + |15.6 - 15.6461| + |23.9 - 20.27858| + |138.3 - 132.9004| = \frac{1}{5} \times \times 17.23538 = 3.45$.

2) MAE-ZOA $= \frac{1}{5} \times |8.4 - 2.98328| + |98.7 - 106.6681| + |15.6 - 14.44949| + |23.9 - 19.74091| + |138.3 - 133.7306| = \frac{1}{5} \times 23.26382 = 4.65$.

3) MAE-PDO $= \frac{1}{5} \times |8.4 - 1.656988| + |98.7 - 100.9093| + |15.6 - 14.95078| + |23.9 - 20.80425| + |138.3 - 123.5154| = \frac{1}{5} \times 27.481882 = 5.5$.

4) MAE-WSO $= \frac{1}{5} \times |8.4 - 4.819038| + |98.7 - 105.0444| + |15.6 - 14.89492| + |23.9 - 19.39078| + |138.3 - 135.7797| = \frac{1}{5} \times 17.659962 = 3.53$.

5) MAE-MFO $= \frac{1}{5} \times |8.4 - 4.515002| + |98.7 - 101.6651| + |15.6 - 15.38723| + |23.9 - 20.11632| + |138.3 - 129.7965| = \frac{1}{5} \times 19.350048 = 3.87$.

D) The Mean Magnitude Relative Error (MMRE)

According to Equation (4)

1) MMRE-GWO $= \frac{1}{5} \times \frac{|8.4-5.02114|}{8.4} + \frac{|98.7-103.4894|}{98.7} + \frac{|15.6-15.6461|}{15.6} + \frac{|23.9-20.27858|}{23.9} + \frac{|138.3-132.9004}{138.3} = \frac{1}{5} \times 0.6442916992 = 0.13$.

2) MMRE-ZOA $= \frac{1}{5} \times \frac{|8.4-2.98328|}{8.4} + \frac{|98.7-106.6681|}{98.7} + \frac{|15.6-14.44949|}{15.6} + \frac{|23.9-19.74091|}{23.9} + \frac{|138.3-133.7306|}{138.3} = \frac{1}{5} \times 1.0063890272 = 0.20$.

3) MMRE-PDO $= \frac{1}{5} \times \frac{|8.4-1.656988|}{8.4} + \frac{|98.7-100.9093|}{98.7} + \frac{|15.6-14.95078|}{15.6} + \frac{|23.9-20.80425|}{23.9} + \frac{|138.3-123.5154|}{138.3} = \frac{1}{5} \times 1.1031718572 = 0.22$.

4) MMRE-WSO $= \frac{1}{5} \times \frac{|8.4-4.819038|}{8.4} + \frac{|98.7-105.0444|}{98.7} + \frac{|15.6-14.89492|}{15.6} + \frac{|23.9-19.39078|}{23.9} + \frac{|138.3-135.7797|}{138.3} = \frac{1}{5} \times 0.7426757914 = 0.15$.

5) MMRE-MFO $= \frac{1}{5} \times \frac{|8.4-4.515002|}{8.4} + \frac{|98.7-101.6651|}{98.7} + \frac{|15.6-15.38723|}{15.6} + \frac{|23.9-20.11632|}{23.9} + \frac{|138.3-129.7965|}{138.3} = \frac{1}{5} \times 0.7259792754 = 0.15$.

E) The Root Mean Squared Error (RMSE)

According to Equation (5)

1)$RMSE - GWO = \sqrt{15.33} = 3.91$.

2)$RMSE - ZOA = \sqrt{26.47} = 5.14$.

3)$RMSE - PDO = \sqrt{55.79} = 7.47$.

4)$RMSE - WSO = \sqrt{16.05} = 4.01$.

5)$RMSE - MFO = \sqrt{22.11} = 4.70$.

F)The R-Squared (R^2)

According to Equation (6)

$Mean(act) = \frac{(8.4+98.7+15.6+23.9+138.3)}{5} = 56.98$.

1) $R^2 - GWO = ((8.4-56.98)^2 + (98.7-56.98)^2 + (15.6-56.98)^2 + (23.9-56.98)^2 + (138.3-56.98)^2) - 76.62753545)/((8.4-56.98)^2 + (98.7-56.98)^2 + (15.6-56.98)^2 + (23.9-56.98)^2 + (138.3-56.98)^2) = \frac{(13520.108-76.62753545)}{13520.108} = 0.9943$.

2) $R^2 - ZOA = ((8.4-56.98)^2 + (98.7-56.98)^2 + (15.6-56.98)^2 + (23.9-56.98)^2 + (138.3-56.98)^2) - 132.3325924)/((8.4-56.98)^2 + (98.7-56.98)^2 + (15.6-56.98)^2 + (23.9-56.98)^2 + (138.3-56.98)^2) = \frac{(13520.108-132.3325924)}{13520.108} = 0.9902$.

3) $R^2 - PDO = ((8.4-56.98)^2 + (98.7-56.98)^2 + (15.6-56.98)^2 + (23.9-56.98)^2 + (138.3-56.98)^2) - 278.9387692)/((8.4-56.98)^2 + (98.7-56.98)^2 + (15.6-56.98)^2 + (23.9-56.98)^2 + (138.3-56.98)^2) = \frac{(13520.108-278.9387692)}{13520.108} = 0.9794$.

4) $R^2 - WSO = ((8.4-56.98)^2 + (98.7-56.98)^2 + (15.6-56.98)^2 + (23.9-56.98)^2 + (138.3-56.98)^2) - 80.2568151102)/((8.4-56.98)^2 + (98.7-56.98)^2 + (15.6-56.98)^2 + (23.9-56.98)^2 + (138.3-56.98)^2) = \frac{(13520.108-80.2568151102)}{13520.108} = 0.9941$.

5) $R^2 - MFO = ((8.4-56.98)^2 + (98.7-56.98)^2 + (15.6-56.98)^2 + (23.9-56.98)^2 + (138.3-56.98)^2) - 110.5560451/((8.4-56.98)^2 + (98.7-56.98)^2 + (15.6-56.98)^2 + (23.9-56.98)^2 + (138.3-56.98)^2) = \frac{13520.108-110.5560451}{13520.108} = 0.9918$.

Table 22. First, using the VAF as an evaluation metric, the VAF of GWO is 99.52%, which is the maximum value found when compared with ZOA, PDO, WSO, MFO, and the models' values proposed by [76]. The VAF values for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 98.63%, 97.60%, 98.70%, 99.101%, 98.72%, 99.44%, and 99.45%, respectively.

Second, by using the MSE as an evaluation metric, the MSE of GWO is 15.33, and the MSE for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 45.02, 114.79, 52.85, 26.47, 55.79, 16.05, and 22.11, respectively. These values indicate that the GWO model can reduce 29.69, 99.46, 37.52, 11.14, 40.46, 0.72, and 6.78 errors, respectively.

Third, by using the MAE as an evaluation metric, the MAE of GWO is 3.45, and the MAE for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 5.57, 7.83, 5.29, 4.65, 5.5, 3.53, and 3.87, respectively. These values indicate that the GWO model can reduce the errors by 2.12, 4.38, 1.84, 1.2, 2.05, 0.08, and 0.42.

Fourth, the MMRE of GWO is 0.13, and for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 0.24, 0.27, 0.21, 0.20, 0.22, 0.15, and 0.15, respectively. This means that the GWO can reduce the errors by reducing 0.11, 0.14, 0.08, 0.07, 0.09, 0.02, and 0.02, respectively.

Fifth, the RMSE of GWO is 3.91, and for Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 6.62, 9.86, 7.19, 5.14, 7.47, 4.01, and 4.70, respectively. This means the GWO can reduce the errors by 2.71, 5.95, 3.28, 1.23, 3.56, 0.1, and 0.79, respectively.

Sixth, it is observed that the GWO has the highest value among all other models. The R^2 of GWO is 0.9943%, while the Firefly, GA, PSO, ZOA, PDO, WSO, and MFO models are 0.9833, 0.9575, 0.9805, 0.9902, 0.9794, 0.9941, and 0.9918, respectively.

It can be concluded that the GWO algorithm obtained the highest value in the VAF and R^2 and the lowest value in relative errors. Therefore, the COCOMO Model II proposed by [67] as an extension 2 of the basic COCOMO software project effort, should be estimated with the new parameter values generated by the GWO model.

After the GWO proved its efficiency in the three COCOMO-based models against other algorithms, and when compared with the three proposed models, which resulted from the employment of the GWO in each model, i.e., the basic-COCOMO, and another two models COCOMO Model I, COCOMO Model II.

It is observed that the COCOMO Model II outperformed the basic COCOMO with four evaluation metrics, which are VAF, MSE, RMSE, and R^2, and it also outperformed the COCOMO Model I with five evaluation metrics: VAF, MSE, MAE, RMSE, and R^2. Therefore, using the COCOMO Model II structure proposed by [67] is highly recommended instead of the basic COCOMO and the COCOMO Model I in estimating the value of software effort, but using the improved optimized parameters using the GWO in Table 20.

The COCOMO Model has been widely used in many real-world applications, proving its usefulness in estimating and administrating industry projects, government initiatives, aerospace and defense, enterprise solutions, healthcare software, gaming industry, web and mobile, and infrastructure software.

VIII. CONCLUSION AND FUTURE WORK

The estimated effort exerts a significant impact on cost estimation. In the area of software engineering, a lot of scholars have endeavored to introduce methods and models for precisely estimating effort. Accurate effort estimation enables us to finish projects on schedule and within budget.

In this study, the Grey Wolf Optimization is employed to enhance the coefficients' values of the basic-COCOMO, Model I, and Model II found by [67]. Also, the results were compared against the Firefly Algorithm, Genetic Algorithm, and PSO Algorithm. Moreover, to check the efficiency of the GWO in finding the optimal value of effort estimation, four other algorithms are applied, including ZOA, PDO, WSO, and MFO. Moreover, a comparison has been carried out to compare the results with the GWO results. VAF, MSE, MAE, MMRE, RMSE, and R^2 evaluation metrics are employed to evaluate the optimized models. The results established the efficiency of the GWO in finding the optimal value of effort estimation over other meta-heuristic algorithms.

In the future, this work can be extended to optimize the effort estimation models provided by M. Uysal and the intermediate COCOMO Model. Moreover, another advanced meta-heuristic algorithm, which may show more effectiveness than the GWO, can be used to develop more efficient models for software effort estimation. The results can be compared with the GWO. It is also recommended to improve the results, whether modifying the existing algorithms or hybridizing the GWO algorithm with other algorithms. Meanwhile, the hybrid approach of multiple algorithms increases the probability of finding the optimal solution efficiently and rapidly. Besides, it is helpful to concentrate on using the GWO algorithm in tuning parameters for other effort estimation models. Another suggested work to be carried out in the future is the optimization of COQUAMO model parameters. Also, some statistical tests, such as Wilcoxon and ANOVA, could be performed to ensure the quality of the proposed models.

ACKNOWLEDGMENT

The authors would like to thank the University of Science and Technology for their provided facilities, which helped to improve the quality of this work.

REFERENCES

[1] P. S. Sandhu, P. Bassi, and A. S. Brar, "Software effort estimation using soft computing techniques," *World Acad. Sci., Eng. Technol.*, vol. 46, p. 2008, Dec. 2008.

[2] S. Shekhar and U. Kumar, "Review of various software cost estimation techniques," *Int. J. Comput. Appl.*, vol. 141, no. 11, pp. 31–34, May 2016.

[3] L. Nerkar and P. Yawalkar, "Software cost estimation using algorithmic model and non-algorithmic model a review," *Int. J. Comput. App.*, vol. 2, pp. 4–7, 2014.

[4] S. Aljahdali and A. F. Sheta, "Software effort estimation by tuning COOCMO model parameters using differential evolution," in *Proc. ACS/IEEE Int. Conf. Comput. Syst. Appl. (AICCSA)*, May 2010, pp. 1–6.

[5] A. Abu-Srhan, A. Sleit, and A. Sharieh, "Parameters estimation of the COCOMO model using hybrid algorithm of genetic algorithm and cuckoo search algorithm," *Proc. New Trends Inf. Technol.*, pp. 67–73, Apr. 2017.

[6] A. Verma and Preeti, "Calibrating intermediate COCOMO model using genetic algorithm," in *Proc. Int. Conf. Comput., Commun., Intell. Syst. (ICCCIS)*, Feb. 2021, pp. 174–179.

[7] C. A. U. Hassan, M. S. Khan, R. Irfan, J. Iqbal, S. Hussain, S. S. Ullah, R. Alroobaea, and F. Umar, "Optimizing deep learning model for software cost estimation using hybrid meta-heuristic algorithmic approach," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–20, Oct. 2022.

[8] A. A. Fadhil and R. G. Alsarraj, "Exploring the whale optimization algorithm to enhance software project effort estimation," in *Proc. 6th Int. Eng. Conf. Sustainable Technol. Develop. (IEC)*, Feb. 2020, pp. 146–151.
[9] A. Puspaningrum, F. P. B. Muhammad, and E. Mulyani, "Flower pollination algorithm for software effort coefficients optimization to improve effort estimation accuracy," *JUITA, Jurnal Informatika*, vol. 9, no. 2, pp. 139–144, Nov. 2021.
[10] K. R. Shweta, "Software cost and effort estimation using ensemble duck traveler optimization algorithm (eDTO) in earlier stage," *Turkish J. Comput. Math. Educ.*, vol. 12, pp. 3300–3311, Jun. 2021.
[11] B. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981.
[12] R. K. Sachan and D. S. Kushwaha, "Anti-predatory NIA based approach for optimizing basic COCOMO model," in *Proc. 10th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence)*, Jan. 2020, pp. 710–715.
[13] M. S. Khan, C. A. U. Hassan, M. A. Shah, and A. Shamim, "Software cost and effort estimation using a new optimization algorithm inspired by strawberry plant," in *Proc. 24th Int. Conf. Autom. Comput. (ICAC)*, Sep. 2018, pp. 1–6.
[14] R. K. Sachan, A. Nigam, A. Singh, S. Singh, M. Choudhary, A. Tiwari, and D. S. Kushwaha, "Optimizing basic COCOMO model using simplified genetic algorithm," *Proc. Comput. Sci.*, vol. 89, pp. 492–498, Jan. 2016.
[15] M. Algabri, F. Saeed, H. Mathkour, and N. Tagoug, "Optimization of soft cost estimation using genetic algorithm for NASA software projects," in *Proc. 5th Nat. Symp. Inf. Technol., Towards New Smart World (NSITNSW)*, Feb. 2015, pp. 1–4.
[16] F. S. Alaa and A. Al-Afeef, "A GP effort estimation model utilizing line of code and methodology for NASA software projects," in *Proc. 10th Int. Conf. Intell. Syst. Design Appl.*, Nov. 2010, pp. 290–295.
[17] D. Nandal and O. Sangwan, "Software cost estimation by optimizing COCOMO model using hybrid BATGSA algorithm," *Int. J. Intell. Eng. Syst.*, vol. 11, no. 4, pp. 250–263, Aug. 2018.
[18] S. M. S. Jafari and F. Ziaaddini, "Optimization of software cost estimation using harmony search algorithm," in *Proc. 1st Conf. Swarm Intell. Evol. Comput. (CSIEC)*, Mar. 2016, pp. 131–135.
[19] A. Sharma and N. Chaudhary, "Software cost estimation for Python projects using genetic algorithm," in *Proc. ICCIS*, 2020, pp. 137–148.
[20] J. Razmi, R. Ghodsi, and M. Jokar, "Cost estimation of software development: Improving the COCOMO model using a genetic algorithm approach," *Int. J. Manage. Pract.*, vol. 3, no. 4, pp. 346–368, 2009.
[21] C. A. U. Hassan and M. S. Khan, "An effective nature inspired approach for the estimation of software development cost," in *Proc. 16th Int. Conf. Emerg. Technol. (ICET)*, Dec. 2021, pp. 1–6.
[22] M. Padmaja and D. Haritha, "Software effort estimation using grey relational analysis," *Int. J. Inf. Technol. Comput. Sci.*, vol. 9, no. 5, pp. 52–60, May 2017.
[23] H. Patra and K. Rajnish, "A new empirical model to increase the accuracy of software cost estimation," *Int. J. Eng.*, vol. 30, no. 10, pp. 1487–1493, Oct. 2017.
[24] M. Kaur, "Estimation of effort using nature inspired optimization techniques," *Int. J. Acad. Res. Dev.*, vol. 3, no. 1, pp. 197–199, Jan. 2018.
[25] M. H. Saadi, V. K. Bardsiri, and F. Ziaaddini, "The application of meta-heuristic algorithms to improve the performance of software development effort estimation models," *Int. J. Appl. Evol. Comput.*, vol. 6, no. 4, pp. 39–68, Oct. 2015.
[26] R. Jain, V. K. Sharma, and S. Hiranwal, "Reduce mean magnitude relative error in software cost estimation by HOD-COCOMO algorithm," in *Proc. Int. Conf. Control, Instrum., Commun. Comput. Technol. (ICCICCT)*, Dec. 2016, pp. 708–712.
[27] A. F. Sheta, D. Rine, and S. Kassaymeh, "Software effort and function points estimation models based radial basis function and feedforward artificial neural networks," *Int. J. Next-Generation Comput.*, vol. 6, pp. 192–205, 2015.
[28] E. R. Avasthy, B. Batra, and H. Kundra, "A hybrid parametric model for enrichment of software effort estimation," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 10, 2016.
[29] S. M. Padmaja and D. Haritha, "Software effort estimation using meta heuristic algorithm," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, 2017.
[30] S. Kumari and S. Pushkar, "Software cost estimation using cuckoo search," in *Proc. Int. Conf. Comput. Intell.*, 2017, pp. 167–175.
[31] A. Wadhwa, S. Jain, and C. Gupta, "An effective precision enhancement approach to estimate software development cost: Nature inspired way," *J. Telecommun., Electron. Comput. Eng.*, vol. 9, no. 3, pp. 85–91, 2017.
[32] O. Benediktsson and D. Dalcher, "Effort estimation in incremental software development," *IEE Proc.-Softw.*, vol. 150, no. 6, pp. 351–357, 2003.
[33] A. Sheta, D. Rine, and A. Ayesh, "Development of software effort and schedule estimation models using soft computing techniques," in *Proc. IEEE Congr. Evol. Comput., IEEE World Congr. Comput. Intell.*, Jun. 2008, pp. 1283–1289.
[34] J. Salt, "A GP effort estimation model utilizing line of code and methodology for NASA software projects," IEEE, New York, NY, USA, Tech. Rep., 2010.
[35] A. Galinina, O. Burceva, and S. Parshutin, "The optimization of COCOMO model coefficients using genetic algorithms," *Inf. Technol. Manage. Sci.*, vol. 15, no. 1, pp. 45–51, Jan. 2012.
[36] N. Dewan and S. Sehra, "Ant colony optimization based software effort estimation," *Int. J. Comput. Sci. Technol.*, vol. 5, no. 3, pp. 53–56, 2014.
[37] S. Chalotra, S. K. Sehra, Y. S. Brar, and N. Kaur, "Tuning of COCOMO model parameters by using bee colony optimization," *Indian J. Sci. Technol.*, vol. 8, no. 14, p. 1, Jul. 2015.
[38] A. F. Sheta and A. Al-Afeef, "Software effort estimation for NASA projects using genetic programming," *J. Intell. Comput.*, vol. 1, no. 3, p. 147, Sep. 2010.
[39] T. T. Khuat and M. H. Le, "Optimizing parameters of software effort estimation models using directed artificial bee colony algorithm," *Informatica*, vol. 40, Oct. 2016.
[40] M. Uysal, "Estimation of the effort component of the software projects using simulated annealing algorithm," *World Acad. Sci., Eng. Technol.*, vol. 41, pp. 258–261, 2008.
[41] B. KumarSingh and A. K. Misra, "Software effort estimation by genetic algorithm tuned parameters of modified constructive cost model for NASA software projects," *Int. J. Comput. Appl.*, vol. 59, no. 9, pp. 22–26, Dec. 2012.
[42] N. Sharma, A. Sinhal, and B. Verma, "Software assessment parameter optimization using genetic algorithm," *Int. J. Comput. Appl.*, vol. 72, no. 7, pp. 8–13, Jun. 2013.
[43] P. V. G. D. P. Reddy, "Particle swarm optimization in the fine-tuning of fuzzy software cost estimation models," *Int. J. Softw. Eng.*, vol. 1, no. 2, pp. 12–23, 2010.
[44] P. P. Reddy and C. V. Hari, "Fuzzy based PSO for software effort estimation," in *Proc. Int. Conf. Adv. Inf. Technol. Mobile Commun.*, 2011, pp. 227–232.
[45] P. V. G. D. P. Reddy and C. Hari, "Software effort estimation using particle swarm optimization with inertia weight," *Int. J. Softw. Eng.*, vol. 2, no. 4, pp. 87–96, 2011.
[46] A. Tripathi, K. K. Mishra, S. Tiwari, and N. Kumar, "Improved software cost estimation models: A new perspective based on evolution in dynamic environment," *J. Intell. Fuzzy Syst.*, vol. 35, no. 2, pp. 1707–1720, Aug. 2018.
[47] S. K. T. Ziauddin, K. Zaman, and S. Zia, "Software cost estimation using soft computing techniques," *Adv. Inf. Technol. Manage.*, vol. 2, no. 1, pp. 233–238, 2012.
[48] T. Singh, R. Singh, and K. K. Mishra, "Software cost estimation using environmental adaptation method," *Proc. Comput. Sci.*, vol. 143, pp. 325–332, Jan. 2018.
[49] S. Bhatia, A. Bawa, and V. K. Attri, "A review on genetic algorithm to deal with optimization of parameters of constructive cost model," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 4, pp. 405–408, 2015.
[50] M. Uysal, *Estimation of the Effort Component of the Software Projects Using Heuristic Algorithms*. Rijeka, Croatia: InTech, 2010.
[51] S. Chhabra and H. Singh, "Optimizing design parameters of fuzzy model based COCOMO using genetic algorithms," *Int. J. Inf. Technol.*, vol. 12, no. 4, pp. 1259–1269, Dec. 2020.
[52] I. C. Suherman, R. Sarno, and Sholiq, "Implementation of random forest regression for COCOMO II effort estimation," in *Proc. Int. Seminar Appl. Technol. Inf. Commun. (iSemantic)*, Sep. 2020, pp. 476–481.
[53] A. A. Fadhil, R. G. H. Alsarraj, and A. M. Altaie, "Software cost estimation based on dolphin algorithm," *IEEE Access*, vol. 8, pp. 75279–75287, 2020.
[54] A. A. Fadhil and B. S. Bahnam, "A hybrid model of ant-lion optimization with cuttlefish algorithm for software effort estimation," Kansai Univ., Iraq, Tech. Rep., Nov. 2020.
[55] M. A. Saleem, R. Ahmad, T. Alyas, M. Idrees, A. A. Farooq, A. Shahid, and K. Ali, "Systematic literature review of identifying issues in software cost estimation techniques," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 8, 2019.
[56] B. Khan, R. Naseem, M. Binsawad, M. Khan, and A. Ahmad, "Software cost estimation using flower pollination algorithm," *J. Internet Technol.*, vol. 21, no. 5, pp. 1243–1251, 2020.

[57] P. K. Sethy and S. Rani, "Improvement in COCOMO modal using optimization algorithms to reduce MMRE values for effort estimation," in *Proc. 4th Int. Conf. Internet Things, Smart Innov. Usages (IoT-SIU)*, Apr. 2019, pp. 1–4.
[58] A. Ullah, B. Wang, J. Sheng, J. Long, M. Asim, and Z. Sun, "Optimization of software cost estimation model based on biogeography-based optimization algorithm," *Intell. Decis. Technol.*, vol. 14, no. 4, pp. 441–448, Jan. 2021.
[59] W. D. Sunindyo and C. Rudiyanto, "Improvement of COCOMO II model to increase the accuracy of effort estimation," in *Proc. Int. Conf. Electr. Eng. Informat. (ICEEI)*, Jul. 2019, pp. 140–145.
[60] P. Singal, A. C. Kumari, and P. Sharma, "Estimation of software development effort: A differential evolution approach," *Proc. Comput. Sci.*, vol. 167, pp. 2643–2652, Jan. 2020.
[61] S. P. Singh, "Cost estimation model using enhance-based differential evolution algorithm," *Iran J. Comput. Sci.*, vol. 3, no. 2, pp. 115–126, Jun. 2020.
[62] G. Vats, R. Agarwal, and L. U. Biet, "Software cost estimation model development and parameter optimization using genetic algorithm," *Int. J. Res. Develop. Appl. Sci. Eng.*, 2021.
[63] D. K. K. Reddy and H. S. Behera, "Software effort estimation using particle swarm optimization: Advances and challenges," in *Proc. CIPR*, 2020, pp. 243–258.
[64] P. S. Kumar and H. Behera, "Estimating software effort using neural network: An experimental investigation," in *Proc. CIPR*, 2020, pp. 165–180.
[65] J. W. Bailey and V. R. Basili, "A meta-model for software development resource expenditures," in *Proc. 5th Int. Conf. Softw. Eng.*, 1981, pp. 107–116.
[66] T. Urbanek, Z. Prokopova, R. Silhavy, and V. Vesela, "Prediction accuracy measurements as a fitness function for software effort estimation," *SpringerPlus*, vol. 4, no. 1, pp. 1–17, Dec. 2015.
[67] 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, Feb. 2006.
[68] E. Trojovská, M. Dehghani, and P. Trojovský, "Zebra optimization algorithm: A new bio-inspired optimization algorithm for solving optimization algorithm," *IEEE Access*, vol. 10, pp. 49445–49473, 2022.
[69] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.
[70] S. K. Sahoo, A. K. Saha, A. E. Ezugwu, J. O. Agushaka, B. Abuhaija, A. R. Alsoud, and L. Abualigah, "Moth flame optimization: Theory, modifications, hybridizations, and applications," *Arch. Comput. Methods Eng.*, vol. 30, no. 1, pp. 391–426, Jan. 2023.
[71] A. E. Ezugwu, J. O. Agushaka, L. Abualigah, S. Mirjalili, and A. H. Gandomi, "Prairie dog optimization algorithm," *Neural Comput. Appl.*, vol. 34, no. 22, pp. 20017–20065, Nov. 2022.
[72] M. Braik, A. Hammouri, J. Atwan, M. A. Al-Betar, and M. A. Awadallah, "White shark optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems," *Knowl.-Based Syst.*, vol. 243, May 2022, Art. no. 108457.
[73] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
[74] R. Mohd, M. A. Butt, and M. Z. Baba, "Grey wolf-based linear regression model for rainfall prediction," *Int. J. Inf. Technol. Syst. Approach*, vol. 15, no. 1, pp. 1–18, Oct. 2021.
[75] E. Dada, S. Joseph, D. Oyewola, A. A. Fadele, H. Chiroma, and S. M. Abdulhamid, "Application of grey wolf optimization algorithm: Recent trends, issues, and possible horizons," *Gazi Univ. J. Sci.*, vol. 35, no. 2, pp. 485–504, Jun. 2022.
[76] N. Ghatasheh, H. Faris, I. Aljarah, and R. M. H. Al-Sayyed, "Optimizing software effort estimation models using firefly algorithm," 2019, *arXiv:1903.02079*.

NADA MOHAMMED ALSHEIKH was born in Ibb, Yemen. She received the B.S. degree (Hons.) in information systems specializing in electronic commerce from the University of Science and Technology, in 2015, and the M.S. degree in software engineering from the University of Science and Technology, Aden, Yemen. Since 2017, she has been an Android Developer and Web Designer.

NABIL MOHAMMED MUNASSAR was born in Saudi Arabia, in 1978. He received the B.S. degree in computer science from the University of Science and Technology, Hodeidah Branch, Yemen, in 2001, the M.S. degree in computer information systems from the Arab Academy for Banking and Financial Sciences, Sana'a Branch, Yemen, in 2007, and the Ph.D. degree in computer sciences from Jawaharlal Nehru Technological University, Hyderabad, India, in 2015. Since 2001, he has been a Lecturer with the Faculty of Computers and IT, University of Science and Technology, Yemen. He is the author of more than 15 articles and has many funded research projects. His research interests include information technology, software engineering, databases, system analysis, and artificial intelligence.