Cluster Computing (2024) 27:737–760
https://doi.org/10.1007/s10586-023-03979-y

Software effort estimation modeling and fully connected artificial neural network optimization using soft computing techniques

Sofian Kassaymeh[1] · Mohammed Alweshah[1,2] · Mohammed Azmi Al-Betar[3,4] · Abdelaziz I. Hammouri[2] · Mohammad Atwah Al-Ma'aitah[5]

Received: 12 September 2022 / Revised: 1 December 2022 / Accepted: 20 January 2023 / Published online: 12 February 2023

Abstract

In software engineering, the planning and budgeting stages of a software project are of great importance to all stakeholders, including project managers as well as clients. The estimated costs and scheduling time needed to develop any software project before and/or during startup form the basis of a project's success. The main objective of soft- ware estimation techniques is to determine the actual effort and/or time required for project development. The use of machine learning methods to address the estimation problem has, in general, proven remarkably successful for many engineering problems. In this study, a fully connected neural network (FCNN) model and a metaheuristic, gray wolf optimizer (GWO), called GWO-FC, is proposed to tackle the software development effort estimation (SEE) problem. The GWO is integrated with FCNN to optimize the FCNN parameters in order to enhance the accuracy of the obtained results by improving the FCNN's ability to explore the parameter search field and avoid falling into local optima. The proposed technique was evaluated utilizing various benchmark SEE datasets. Furthermore, various recent algorithms from the literature were employed to verify the GWO-FC performance. In terms of accuracy, comparative outcomes reveal that the GWO-FC performs better than other methods in most datasets and evaluation criteria. Experimental outcomes reveal the strong potential of the GWO-FC method to achieve reliable estimation results.

Keywords Metaheuristic · Optimization · Grey wolf optimizer · Fully-connected neural network · Software development effort estimation

1 Introduction

The process of estimating software development efforts prior to and/or during the development stage is critical to the success of a software project and to reducing risk. Software projects do not have the same structure and nature and so the estimation of the effort process may become a challenging task [1, 2]. In the literature, several machine learning (ML) methods have been proposed to enhance software development effort estimation (SEE) [3–7]. Furthermore, a wide variety of research articles have been

✉ Sofian Kassaymeh
s.kassaymeh@aut.edu.jo; samsaak@gmail.com

Mohammed Alweshah
weshah@bau.edu.jo

Mohammed Azmi Al-Betar
m.albetar@ajman.ac.ae; mohbetar@bau.edu.jo

Abdelaziz I. Hammouri
aziz@bau.edu.jo

Mohammad Atwah Al-Ma'aitah
dr-moh1975@bau.edu.jo

1 Software Engineering Department, Faculty of Information Technology, Aqaba University of Technology, Aqaba, Jordan

2 Department of Computer Science, Prince Abdullah Bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University, Salt, Jordan

3 Artificial Intelligence Research Center (AIRC), College of Engineering and Information Technology, Ajman University, Ajman, United Arab Emirates

4 Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, Irbid, Jordan

5 MIS Department, Amman University College, Al-Balqa Applied University, Amman, Jordan

published on the optimization of the parameters of the three COCOMO-based models by employing an artificial neural network (ANN) [8–11].

Commonly, ANNs have been employed to tackle software estimation problems due to their suitability for arbitrary accuracy and superior predictive ability [12, 13]. They have also been used in various real-world applications [10, 14–16], proving capable of tackling estimation issues very effectively because of their learning capacity. The ANN learning technique mainly involves the updating of the weights and biases of neurons as they modify the transmission of signals among interconnected neurons. Therefore, the learning technique defines how the weights and biases should be updated in order, for example, to optimize the loss function.

Several studies have employed various types of ANNs to address SEE problems. For instance, [17] compared four previous studies [18–21] which used different neural network architectures–multilayer perceptron (MLP), a general regression neural network (GRNN), a radial basis function neural network (RBF), and cascade correlation neural networks (CCNN)—to estimate software project efforts. In their study, the authors of [17] found that each network outperformed the previous one, confirming the ability of different types of networks to address the SEE problem but with varying capabilities. One of the most popular and efficient learning techniques employed to train an ANN is the backpropagation algorithm, which has proved able to address a variety of estimation and classification issues [22].

The backpropagation algorithm utilizes a local search-based technique called gradient descent to minimize the value of the error function in the weight space [23]. The fully connected neural network (FCNN) has many superior properties, including an excellent self-learning technique, robustness, self-adaptation, and generalization capacity [24]. In addition, a three-layer FCNN can tackle non-linear functions [25]. The FCNN can also be applied to a wide range of research fields, such as function approximation, image processing, and pattern recognition [26]. Nevertheless, the FCNN suffers from many drawbacks, such as slow convergence speed [27–29], becoming easily stuck in local optima [27, 28], low convergence accuracy [28], and high dependency on initial parameters [29]. To overcome these drawbacks, several metaheuristic algorithms have been proposed, most notably the genetic algorithm (GA) [30], particle swarm optimization (PSO) [31], the artificial bee colony (ABC) [32], the whale optimization algorithm [33], biogeography-based optimization (BBO) [34], the firefly optimization algorithm (FFA) [35], the bat algorithm (BA) [36], and the cuckoo search (CS) [37].

In this study, we propose a novel technique known as GWO-FC, in which the gray wolf optimizer (GWO) [38, 39] is integrated with the FCNN to optimize the FCNN parameters (i.e., weights and biases) so that they are more sensitive to tackling the SEE problem. It is essential to find a low-complexity and high-utility estimation method. In this regard, the GWO is fast, robust, and has simple features [40] which support dependability. The motivation for this work is the priority that must be given to managing the expenditure and effort incurred during the software project development cycle. The effort estimation process aims to provide an accurate estimation of the cost of software development, as well as assist in the efficient use and allocation of human and computational resources for development tasks.

To validate the SEE findings obtained by the GWO-FC approach, 12 dataset instances for the SEE were selected from different repositories, such as PROMISE and GitHub. First, the data preparation step for the selected datasets with parameter configurations was conducted. Then, the performance of the proposed GWO-FC was studied and analyzed in terms of convergence behavior. After this, a statistics-based evaluation was performed to compare the GWO-FC against traditional FCNN methods. Finally, the efficiency of the proposed GWO-FC was further validated by comparing its results with those of selected state-of-the-art methods. The analysis of the findings shows that the GWO-FC approach is a viable method for the SEE problem in the field of software engineering.

It should be noted that the gray wolf optimizer (GWO) has been used in previous studies to tackle the problem addressed in this study [37, 41]. In [41], the authors used the GWO algorithm to address the shortcomings of conventional software prediction methods, a result of imprecise model construction and erroneous outcomes. They combined three metaheuristic algorithms–the GWO, harmony search (HSA), and strawberry algorithms (SB)–to optimize the COCOMO effort estimation method and applied the developed model to a NASA dataset. Their study did not use ML methods with the GWO to address the prediction efforts problem, and instead they opted for outdated traditional methods. In [37], a combination of the GWO and SB algorithms was utilized to build a parametric model for the SEE problem; GWO was used to optimize the weights of a deep neural network, while SB was used to improve its learning rate. However, their model suffers from a high convergence time as well as a poor balance between exploitation and exploration.

In short, the ultimate goal of this study is to integrate the GWO into the FCNN to tackle the SEE problem. To achieve this, three novel contributions are made in the following order:

- Introduction of the FCNN network to tackle the SEE problem;

- Use of the GWO algorithm as a learning technique for FCNN network to identify best parameter values, consequently enhancing the estimation ability;
- Formulation of the FCNN parameters as input solutions for the GWO;
- Use of the GWO to find the optimal vector to use as the optimal parameter for the FCNN;
- Evaluation of the proposed GWO-FCNN using several well-known benchmark datasets.

The remainder of this article is organized as follows: Sect. 2 provides the background to the study and an overview of the GWO and FCNN, and Sect. 2.3 presents the SEE problem. The proposed method is discussed in Sect. 3, while Sect. 4 presents the experimental results and performance evaluation. Sect. 6 concludes the article.

2 Background

In this section, the GWO is thoroughly discussed and the mathematical formulation of the FCNN presented.

2.1 Grey wolf optimizer

Mirjalili et al. [38] developed the GWO as a population-based metaheuristic algorithm inspired by the social leadership and hunting behavior of a pack of gray wolves. In the GWO, three dominant leaders, α, β, and δ, can lead the remainder of the pack, called ω, to the candidate regions to discover the global solution. The hunting mechanism consists of three stages: encircling, hunting, and attacking the prey.

Encircling: As seen in Eqs. 1 and 2, it is possible to mimic how wolves might surround their prey:

$$D = |C * X_p(t) - X(t)| \tag{1}$$

$$X(t+1) = X_p(t) - A * D \tag{2}$$

where, the prey position is symbolized by X_p, the position vector of a gray wolf is symbolized by X, the current iteration is symbolized

$$A = 2 * A * r_1 - a(t) \tag{3}$$

$$C = 2 * r_2 \tag{4}$$

where, r_1 and r_2 coefficients are vectors with random values ranging from 0 to 1. vector a items is linearly decreased in [2,0] through the iterations using Eq. 5:

$$a(t) = 2 - \frac{(2 * t)}{MaxIter} \tag{5}$$

Hunting: In order to model the wolves' hunting behavior mathematically, it is presumed that α, β, and δ have better knowledge of the prey's location. Therefore, given the location of the three best solutions α, β, and δ, the other wolves ω are forced to follow. The description of the hunting behavior is represented by Eqs. 6, 7, and 8:

$$\begin{aligned} D_\alpha &= |C_1 * X_\alpha - X(t)| \\ D_\beta &= |C_2 * X_\beta - X(t)| \\ D_\delta &= |C_3 * X_\delta - X(t)| \end{aligned} \tag{6}$$

where the coefficients C_1, C_2, and C_3 are computed by Eq. 4.

$$\begin{aligned} X_{i1}(t) &= X_\alpha(t) - A_{i1} * D_\alpha(t) \\ X_{i2}(t) &= X_\beta(t) - A_{i2} * D_\beta(t) \\ X_{i3}(t) &= X_\delta(t) - A_{i3} * D_\delta(t) \end{aligned} \tag{7}$$

where the coefficients X_α, X_β, and X_δ are the first three best solutions at iteration t, while $A1$, $A2$, and $A3$ are computed by Eq. 3, and D_α, D_β, and D_δ are computed by Eq. 6.

$$X(t+1) = \frac{X_{i1}(t) + X_{i2}(t) + X_{i3}(t)}{3} \tag{8}$$

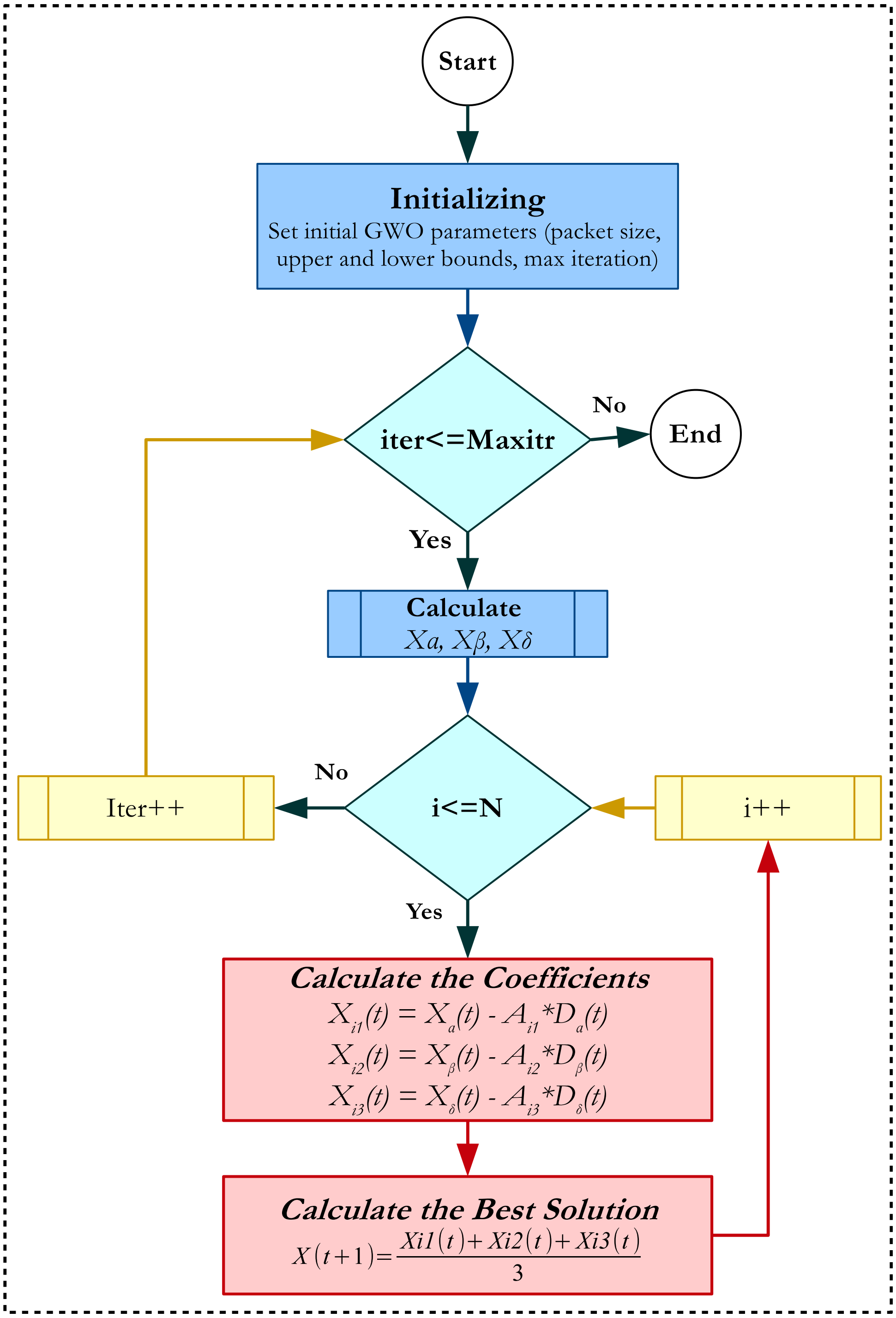
Attacking: Hunting ends when the prey stops and the wolves attack. All of this can be simulated mathematically with the linear decrement in a value over the course of the iterations in order to control exploration and exploitation. Eq. 5 shows that the value of a is updated in each iteration across the range [2,0]. Emary et al. [42] recommend that 50% of the iterations are used for exploration and the remaining iterations for exploitation in a seamless transition. At these moments, wolves randomly move location to any other location in the range between the current one and that of the prey.

Figure 1 depicts a full flowchart of the GWO. Generally, within the search space, the method begins with an initial random formation of wolves. Next, the fitness of each solution (wolves' positions) is then evaluated. The remaining steps are repeated until the halting requirement is met. The maximum number of iterations is defined as the halting requirement. In each iteration, the highest ranked solutions (i.e., wolves α, β, and δ) with the best finesses are considered. Subsequently, the location of each wolf is updated in the above stages (i.e., encircling, hunting, and attacking). Through repetition of the above three stages, the best prey position can be determined, which is α's position.

2.2 Fully connected neural network

One of the most well-liked ANN models is the FCNN. It is widely used to tackle regression and classification problems [43, 44]. Essentially, the FCNN contains several layers in addition to processing elements called neurons.

Fig. 1 Flowchart of the GWO algorithm


Start
Initializing
Set initial GWO parameters (packet size, upper and lower bounds, max iteration)
iter<=Maxitr
No
End
Yes
Calculate
Xa, Xβ, Xδ
No
Iter++
i<=N
i++
Yes
Calculate the Coefficients
X_i1(t) = X_a(t) - A_i1*D_a(t)
X_i2(t) = X_β(t) - A_i2*D_β(t)
X_i3(t) = X_δ(t) - A_i3*D_δ(t)
Calculate the Best Solution
X(t+1) = (Xi1(t)+Xi2(t)+Xi3(t))/3

The layers are stacked parallel to each other, with neurons distributed over each layer. Also, neurons are fully connected to each other between the layers, as shown in Fig. 2. The input layer is the first layer, in which the network receives its input variables, while the output layer is the last. The layers between the input and output layers are known as hidden layers.

Weights are associated with all neuronal connections, which determine the impact of the relevant inputs on neurons. In addition, there is an activation and aggregation function within each neuron to produce the output, with the activation function being unique within a single layer. The aggregation function is shown in Eq. 9, which computes the inputs' weighted sum. The activation functions

(Eq. 14) apply a threshold to the derived weighted sum to produce the neuron's output.

$$net_j = \sum_{i=1}^{n} w_{ij} * x_i + b_j \tag{9}$$

where, the variable of the input i is denoted as x_i, the value of j^{th} neuron bias is denoted as b_j, the value of connection weight from the i^{th} input to the j^{th} neuron is denoted by w_{ij}.

To calibrate the network connection weights, the FCNN training technique is employed, consisting of the forward propagation (FP) and backward propagation (BP) stages respectively. Data are fed to the input layer and then transferred to the output layer after being transferred to the hidden layer in the FP stage. During this pass, the aggregation and activation functions update the weights and biases of each neuron. The estimation error e is often measured in the output layer by computing the difference between the real and anticipated outputs. The obtained error is then back-propagated to the hidden layer in the BP stage to adjust the weights and biases according to the value of e so that the error is reduced. The weights ω_{ij} and ω_{jk} are updated as in Eqs.10 and 11, respectively. In addition, biases b_1 and b_2 are updated as in Eqs.12 and 13, respectively. These two processes are repeated iteratively till the e value approaches zero or a tolerable limit. Thus, the FCNN is trained to reduce the overall network error, which might be viewed as an issue of optimization [45]. The training technique is portrayed in Fig. 3.

$$\widehat{\omega_{ij}} = \omega_{ij} + \eta H_i(1 - H_j) I_i \sum_{k=1}^{t} \omega_{jk} e_k \tag{10}$$

$$\widehat{\omega_{jk}} = \omega_{jk} + \eta H_j e_k \tag{11}$$

where, adjusted weights obtained are donated by $\widehat{\omega_{ij}}$ and $\widehat{\omega_{jk}}$, original weights are donated by ω_{ij} and ω_{jk}, learning rate is donated by η.

$$\widehat{b_j} = b_j + \eta H_j(1 - H_j) \sum_{k=1}^{t} \omega_{jk} e_k \tag{12}$$

$$\widehat{b_k} = b_k + e_k \tag{13}$$

where, adjusted biases obtained are donated by $\widehat{b_j}$ and $\widehat{b_k}$, original biases are donated by b_j and b_k, the learning rate is donated by η.

2.3 The software development effort estimation issue

The estimation of software development effort can be defined as the process of estimating the practical amount of effort necessary to develop a software project from

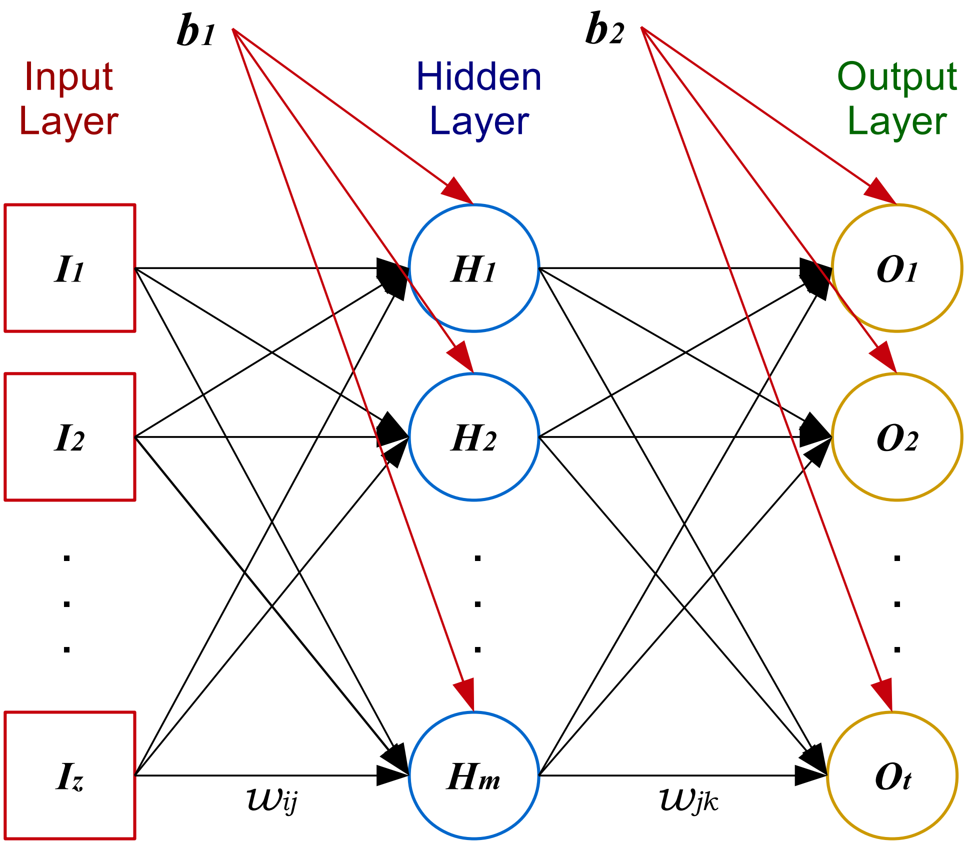


Fig. 2 Basic structure of fully-connected neural network

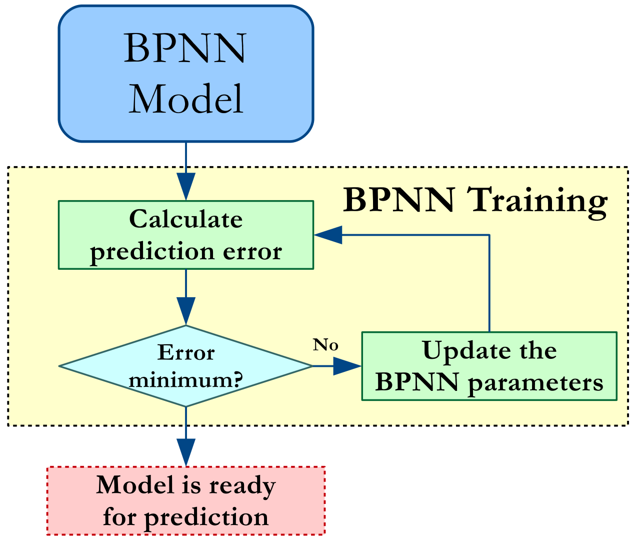

BPNN Model
BPNN Training
Calculate prediction error
Error minimum?
No
Update the BPNN parameters
Model is ready for prediction

Fig. 3 Backpropagation training technique

inconsistent, incomplete, noisy, and uncertain input data [46]. This is the crucial moment when the project manager needs to estimate in advance the substantial resources required for the development of a software project [47]. Suppose there is a reasonable estimation of the effort needed to build a software project. This will facilitate the process of allocating resources to project tasks as well as accurately estimating costs, reducing failures and delays in development, and smoothing the project schedule [10]. The possibility of stakeholders accepting or rejecting a software project is a substantial factor in estimating the effort involved in realizing a software project [4]. In general, underestimation and/or overestimation are the main issues encountered in the forecasting process, with underestimation leading to understaffing of the project, delivery delays, and inaccurate forecasting of budget expenditure. In contrast, overestimation will cause project overrun and loss of resources [48, 49].

Generally, money and person-hour criteria are used to measure effort, which means how many persons per hour spent is needed to develop the software. In [50], there is an explanation of the general factors that may lead to software failure, including the risk of mismanagement, unrealistic software project objectives, employment of unripe technology, inaccurate definition of system requirements, incompetence in managing the complexity of the project, incorrect project status reports, disagreements between stakeholders, labor market pressures and difficulties, and miscommunication between customers/users and software developers. Nevertheless, the success of the software depends mainly on the accuracy of estimating the efforts made to develop it [51]. Consequently, effort estimation must be optimized for a software project because correct estimation is desired by both developers and clients. Furthermore, estimating the effort required assists the developer in building and controlling a software project efficiently, as well as enabling the client to accomplish project contract completion dates, negotiations, and prototype release dates. Although there are many methods of estimating software development effort, it remains difficult for researchers to develop a reliable approach to estimating development efforts.

3 Proposed method

The proposed method involves combining the FCNN and the GWO, as shown in Fig. 4. The FCNN design consists of three layers: an input, output, and single hidden. A single hidden layer is sufficient to enhance the estimation accuracy of the problem addressed in this study [52, 53], and a three-layered FCNN has the ability to address any non-linear functions [54]. In the input layer, the number of neurons is based on the number of dataset features. In the hidden layer, the amount of neurons is established through the trial-and-error method [55]. The total estimated effort is determined by the output layer, which contains only one neuron.

According to Eq.14, the sigmoid function is unique to all neurons in the hidden layer, as well as an aggregation function. In general, most ANN types use the S-shaped sigmoid function because it is thought to be suitable for reducing the influence of overfitting and accelerating the training of the model [56–59]. Likewise, the output layer neurons have a linear activation function, as in Eq.15.

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{14}$$

$$f(x) = x \tag{15}$$

After the FCNN has finished its training process, the vector form of the adjusted parameters (weights and biases) is retrieved and delivered as shown in the following expression and Fig. 5:

$$(\mathbf{w}, \mathbf{b}) = (\mathbf{w}_{1,1}, \mathbf{w}_{1,2}, \ldots, \mathbf{w}_{n,n}, \quad \mathbf{b}_{1,1}, \mathbf{b}_{1,2}, \ldots, \mathbf{b}_{n,n})$$

where. input nodes number referred as n, connection weight is denotes as w_{ij}, bias in the hidden node is represented by b_j.

This vector consists of two parts, as shown in Fig. 5. The first (*blue side*) is for the weights and the second (*red side*) for the biases. After the vector is optimized by the GWO-FC and fed back to the FCNN network, the vector is split into two parts, weights and biases. This returns them to the state/condition they were in before being merged into a single vector, for further use by FCNN.

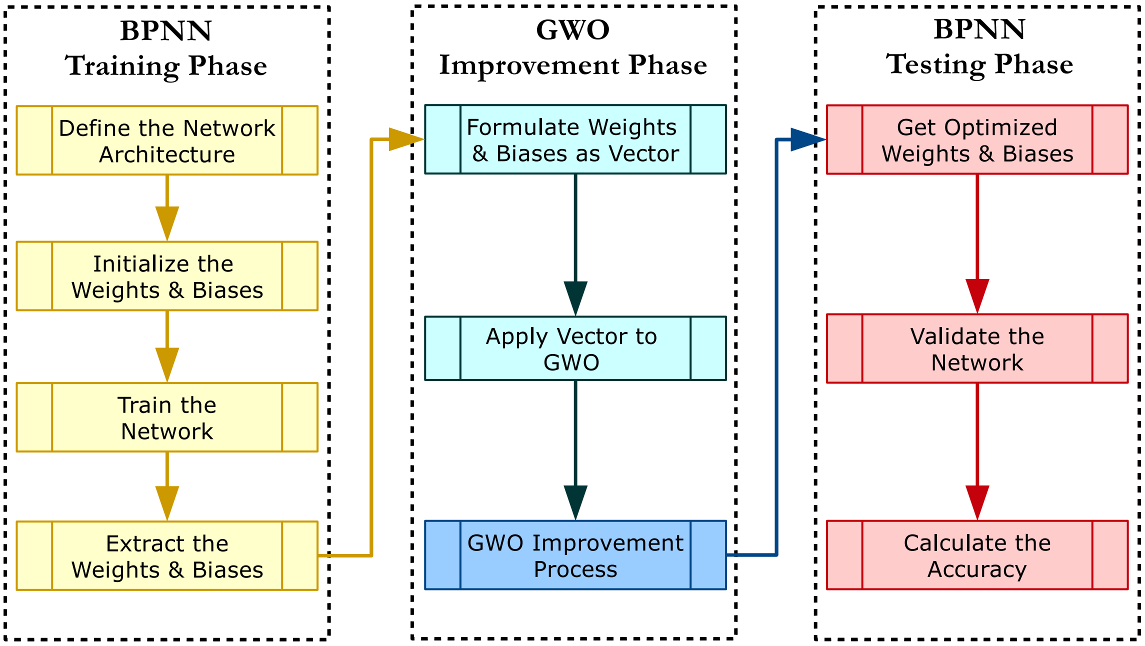

BPNN
Training Phase
Define the Network Architecture
Initialize the Weights & Biases
Train the Network
Extract the Weights & Biases
GWO
Improvement Phase
Formulate Weights & Biases as Vector
Apply Vector to GWO
GWO Improvement Process
BPNN
Testing Phase
Get Optimized Weights & Biases
Validate the Network
Calculate the Accuracy

Fig. 4 Proposed GWO-FC method

The FCNN is trained as far as the population size of the GWO allows, which in this study is equal to 30. With every training process, one vector of FCNN-adjusted weights and biases is modeled, as mentioned earlier. Therefore, all vectors aggregate to form a population that is used as a GWO population. This population is directed to GWO for optimization. The population optimization process is implemented to find the optimal individual solution (vector), which reduces the overall FCNN estimation error, and this process can be considered an optimization problem [45].

The FCNN is a gradient-based search method and so, unfortunately, it has a scaling problem which can lead to a prompt decline in performance when handling high-dimensional issues [45, 60]. Gradient-based search techniques may also become stuck in local minima since FCNN parameters are multi-modal spaces with a range of local minima close to the global minimum [45]. Metaheuristic optimization algorithms can be used to address this problem because these are generally combined with an ANN to obtain high-precision results as well as to minimize network training time [61]. In the present study, the assembled metaheuristic algorithm is the beating heart of the proposed method of using the GWO to optimize the FCNN weights and biases. The main aim of using the GWO algorithm is to reduce the probability of the FCNN falling into local minima. This can be obtained by utilizing the joint process between GWO and FCNN, as the GWO promotes great exploration while the FCNN promotes exploitation, leading to a reasonable balance between them. A previous study has proven that population-based metaheuristic algorithms have powerful exploration capabilities [62].

Returning to the procedure of the proposed method, after the optimal vector has been obtained by the GWO, this vector is returned to the FCNN, which utilizes the vector in the validation process and calculates the accuracy. Mean square error (MSE) is employed in this article as the key fitness function by which to measure output error. The MSE measures the errors between the estimations and the true labels of each estimation process during the training and validation steps of the proposed method, as shown in Eq.16:

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}(A_i - D_i) \tag{16}$$

where, training samples number is donated by m, actual output of the i^{th} instance is donated by A_i, desired output is donated by (D_i).

The reason for using MSE as a primary loss function in this research is due to the tackling of a regression problem, and MSE is a standard measure of test error for this type of problem [63]. In addition, the FCNN model often employs

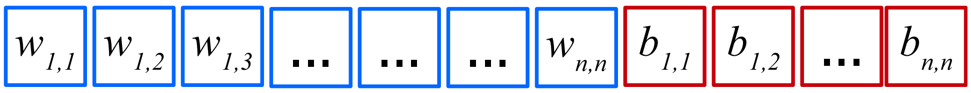


Fig. 5 Single vector (Solution) extracted from FCNN

the MSE as robust loss functions [64, 65]. Furthermore, as seen in Table 3, this study uses several loss functions since there is no single measurement method that is useful for all types of problems [57].

In short, the main contribution of this research is combining the GWO algorithm with FCNN to act as one unit (GWO-FC) to address the SEE problem. Therefore, the parameters (weights and biases) of the BP network are collected as a single vector and sent to the GWO algorithm as input solutions. Then, the GWO optimizes this solution, which is then used as optimal parameters for the BP network. Finally, the developed GWO-FC is evaluated using several global benchmark datasets.

4 Experimental results and performance evaluation

The performance evaluation of the proposed GWO-FC approach to the SEE issue is covered in this section. First, the evaluation criteria, datasets used, experimental design, and parameter composition are discussed. The outcomes of the proposed approach are then contrasted with those of the conventional FCNN on datasets for SEE. After this comparison, the results are presented of a statistical analysis utilizing the Wilcoxon-Mann-Whitney test to generate statistically significant findings. In addition, boxplot and convergence behavior analyses are provided, respectively. Finally, performance validation results are given by comparing the results of the proposed method with those of some state-of-the-art methods when applied to SEE problem datasets.

4.1 Evaluation criteria

The determination of evaluation criteria is critical to the success of a proposed approach. There are different evaluation criteria in the literature and this study sought to utilize the most reliable and common criteria. Heuristic algorithms are stochastic optimization methods which can, therefore, produce different outcomes [45]. Thus, an average of 30 separate runs for each dataset was evaluated to acquire all the experimental outcomes.

MSE served as the primary evaluation criterion in this study, as previously established. Additionally, other criteria were also used, namely: relative absolute error (RAE), mean absolute error (MAE), variance-accounted-for (VAF), Manhattan distance (MD), root mean square error (RMSE), root relative squared error (RRSE), median of magnitude relative error (MdMRE), correlation coefficient (R^2), euclidian distance (ED), standardized accuracy (SA), and effect size ($\triangle$) measures. In utilizing the proposed method, the main aim was to decrease the value of these criteria, except for VAF and R^2, where the aim was to increase their value.

4.2 Datasets used

In this study, twelve different freely and publicly available benchmark datasets were utilized to estimate software development efforts research community. These benchmark datasets have been employed in related works in the literature and thus they are appropriate for use in evaluating the proposed method [3]. The literature states that using a relatively large number of software development effort estimation datasets is helpful for reaching a stable conclusion [66]. All the employed datasets were obtained from GitHub and PROMISE repositories and had several features ranging from 7–27, all of which were used in the experiments. The observations amount ranged between 15–499 and they also had different technological features.

Table 1 provides the following details for each dataset: the name of the dataset, number of features, search space dimensions, estimated time unit, and repository source. It is evident from the table that there is variation in the complexity and size of all datasets. In terms of both the number of features and instances, Albrecht, Kemerer, and Miyazaki are small datasets, whereas China, COCOMO, Maxwell, and NASA are medium/large. The Kemerer, Albrecht, and Kitchenham datasets have the fewest number of features, whereas the COCOMONASA-II and Maxwell have the most. While all other datasets are recorded in person-months, the Maxwell, China, and Desharnais are recorded in person-hours. In all datasets, the dependent variable is effort, expressed in person-months or person-hours.

It is critical to comprehend the scope of software development work to produce an accurate estimate and learn how to deliver estimates for the effort of software development. Estimation is crucial because it enables the developer to determine the expenses and time needed to finish the task. Once it is known how much a person-hour costs and the dependencies of all tasks have been analyzed, one can easily calculate the time it will take to complete the entire software project. A person-hour refers to that portion of work achieved by an average specialist in one hour of uninterrupted work. The term refers to two distinct concepts: *Man (person)* refers to the specialist performing the activity (e.g., analyst, developer, engineer, tester, etc.), and *Hour* refers to 60 minutes of uninterrupted work. Finally, what applies to the term person-hours also applies to person-months, except that the second term is calculated as the total hours worked in a month. The features of the datasets are as follows givn in Table 1.

Table 1 Description of the datasets

Dataset	Features	Dimension	Unit	Source	Reference	Language	Collected from	Available on	How the effort was measured
Albrecht	8	24	Person-months	PROMISE	[67]	COBOL, PL1, DMS		https://rb.gy/egz5md	
China	16	499	Person-hours	PROMISE	[68]		Various software companies	https://rb.gy/uyyobl	
Cosmic	11	42	Person-months	PROMISE		Java,C#.Net,'Java 2 EE', COBOL,VB6,'Oracle eBS', C++, Java, C#, 3GL			
COCOMO81	17	63	Person-months	PROMISE			1981	https://rb.gy/t6ot41	Effort in months (development & management)
COCOMONASA-I	17	60	person-months	PROMISE			NASA projects 1980s and 1990s	https://rb.gy/j3j1p7	Effort in months (development & management)
COCOMONASA-II	2 4	93	person-months	PROMISE			NASA projects 1971–1987	https://rb.gy/83jkcr	Effort in months (development & management)
Desharnais	10	81	Person-hours	GitHub	[69]		Canadian software houses 1989	https://rb.gy/wgfckt	
Kemerer	6	15	Person-months	PROMISE		Cobol, Bliss, Natural		https://rb.gy/x0rx9o	
Kitchenham	7	145	Person-months	PROMISE	[70]		2002	https://rb.gy/pvvfop	
Maxwell	23	62	Person-hours	PROMISE		Four languages	Commercial banks in Finland	https://rb.gy/cugfcz	
Miyazaki 94	8	48	Person-months	PROMISE					
USP05	8	203	Person-months	GitHub		C++, Java, VB, JavaScript, VB Script, SQL, PHP, Perl, ASP, HTML, XML	Unknown	https://rb.gy/y2ppyc	Effort in hours

- Size features: data on the scope of the project as measured by several metrics, e.g., function points (fp), lines of code (loc).
- Environment features: company data, the development team of the project, number and experience of developers, and so on.
- Development features: project technical details, such as the database type and development language employed.
- Project-related features: regarding the project's purpose, type, and requirements.

The IBM DP service corporation generated the Albrecht dataset, which includes 24 samples from industrial IT projects and eight attributes. It is described in terms of KSLOC and FPs, which are weighted sums of inputs, outputs, files, and inquiries for software projects. The China dataset contains information from projects developed by Chinese corporations and considers 16 features and 499 samples. Functional elements are used as independent variables to find how many FPs there are, such as inquiry, file, input, interface, and output.

For the COCOMO dataset created by NASA, this includes 17 features and 63 samples. Among the features are: loc (line of code), rely (reliability of the software), tool (use of software tools), data (size of the datasets), modp (modern programming practices), cplx (process complexity), lexp (language experience), time (cpu time constraint), sced (schedule constraint, stor (main memory constraint), vexp (virtual ma- chine experience), virt (volatility of the machine), pcap (capability of programmers), turn (turnaround time), aexp (application experience), and acap (capability of analysts).

The Kemerer dataset is small, with seven features and 15 samples. There are two category aspects to the independent features (language and hardware). Raw FPs are based on KSLOC, and adjusted function points. The two dependent variables are the project time and overall effort.

The Maxwell dataset contains information about 62 projects, including details of the industrial software initiatives programmed by one of Finland's leading commercial banks. Among the significant independent features are T15 (staff team skill), FPs (SizeFP), T14 (staff tool skills), T01 (customer participation), T13 (staff application knowledge), T02 (development environment adequacy), T12 (staff analysis skills), T03 (staff availability), T11 (installation requirements), T04 (standards used), T10 (efficiency requirements), T05 (methods used), T09 (quality requirements), T06 (tools used), T08 (requirements volatility), and T07 (software logical complexity).

Miyazaki is a medium-sized dataset with 48 samples that include data on projects created by Fujitsu Large Systems Users Group software firms. It has eight independent features, with the dependent variable being the number of person-hours needed to finish the development process from system design through system testing. The number of various record formats, various report forms (form), and various input or output screens (scrn) are all significant dependent variables (file).

In the 1980s, the Desharnais dataset, which contains 81 software projects, was gathered from ten Canadian firms. The total effort was used as a dependent variable in this study, but not the loc. The categorical variables (i.e., language and year end) were also omitted from this study, and the following variables were employed: adjusted FPs, transactions (i.e., the total number of fundamental logical transactions in the system), teamexp (i.e., the team's years of experience), entities (i.e., the number of entities in the system's data model), and managerexp (i.e., the manager's experience measured in years).

The datasets used include data from one or more software firms representing a wide range of application environments and project features. The datasets above have also been utilized in wide ranging practical research to evaluate effort estimation approaches in the literature [71, 72].

4.3 Experimental design

Before the proposed method was assessed, it was necessary to pre-process the data to enable optimal usage of it. By analyzing the variables within the datasets, we found that the characteristics exhibited several notable factors. For instance, large projects are less numerous than smaller ones, which influences the skewness of the data and, as the scale of the project increases, so does the diversity of the effort. There are also some very large data values, which mean that there are outliers. Finally, it appears that the correlations between size and effort varied for different software projects.

Based on the previous findings, it appears that there is a need to use transformation techniques for the data to guarantee that the developed model will traverse the raw data's scale origin. This will take into account the relationships between size and effort, which may be linear, nonlinear, or both. Transforming the data values by applying the transformation technique will bring the data closer to a normal distribution, and also bring the values closer together by reducing larger values to smaller ones. On this basis, the datasets were converted into a new form where all the values were between 0 and 1, in a transformation technique called min-max normalization, as shown in Eq. 17:

$$Y_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{17}$$

where, normalized data is referred as Y_i, data initial value is referred as x_i, minimum and maximum data are referred as x_{max} and x_{min}, respectively.

The second step in the data pre-processing was to divide it into *training* and *testing* sets. The training set accounted for 70% of the original dataset and the testing set 30%. The selection of the data rows for the training set was performed by random selection. The remaining rows were included in the testing set. Random selection was employed to prevent method overfitting and data selection bias. Therefore, for each run of the 30 runs of the proposed method, new training and testing sets of data were formulated randomly. The training phase was executed for all of the datasets used. Then, the testing phase was carried out. Finally, 30 separate runs of each experiment were conducted, so that all the evaluation criteria measurements could be calculated for each dataset by averaging the results obtained for the 30 runs as a final result.

4.4 Parameter configuration

The parameter configuration was the same for all experiments. Where the population size of the GWO was determined to be 30, the maximum iterations of GWO (L) was determined to be 300, and all experiments were performed 30 separate times. Extensive experiments were carried out to find the parameter values (i.e., maximum of iterations and population size), and the optimal values were chosen. The experiments were carried out using a personal computer with an Intel Core i5 processor, Windows 10 system, 8 GB of RAM, 2.0 GHz CPU, using MATLAB 2016a.

4.5 Evaluation comparison of GWO-FC against traditional FCNN

The first experiment was conducted to evaluate the performance of the proposed GWO-FC method. This involved comparing the results obtained by the GWO-FC with those produced by the traditional FCNN for SEE problems on test data. A comparison of the results obtained by the GWO-FC and conventional FCNN for each data set is shown in Tables 2 and 3 in terms of MSE and other measures (i.e., RAE, MAE, VAF (%), MD, RMSE, RRSE, MdMRE, R^2, ED, standardized accuracy (SA) and effect size ($\triangle$) measures), respectively. Table 2 shows that the GWO-FC approach gave more accurate estimates than the classic FCNN in terms of MSE for all datasets. Table 3 shows that the GWO-FC achieved the lowest RRSE, ED, MdMRE, RAE, RMSE, MAE, MD, SA and effect size ($\triangle$) for most datasets. Moreover, the GWO-FC produced the largest VAF and R^2 for all datasets.

The outcomes of the proposed technique thus show that the performance of the FCNN estimator and the quality of the findings were significantly improved by GWO optimization of the FCNN parameters. As previously mentioned, the FCNN tends to become trapped in local optima and has a sluggish convergence rate, and yet the proposed GWO-FC overcame these drawbacks. Providing more optimal weights and biases adds greater balance concerning exploitation and exploration in addition to exponentially accelerating the convergence of the FCNN estimation process. This provides a remarkable generalization estimation performance.

A boxplot was created to show the distribution of the findings obtained using the proposed method. Figure 6 shows the MSE results produced by the GWO-FC method, demonstrating that the majority of the results of the examined datasets were more successful when using the proposed strategy. The effectiveness of the obtained findings may be related to the proposed method's ability to identify the optimal weights of the FCNN, which is essential for resolving the early convergence defect as well as improving the convergence behavior of the FCNN.

4.5.1 Wilcoxon Mann-Whitney statistical test analysis

The significance of the outcomes produced using the suggested strategy was validated with the Wilcoxon-Mann-Whitney statistical test. This test is utilized for the purpose of demonstrating a difference in the value of an ordinal, interval, or ratio variables between two sets. This non-parametric test is utilized for continuous, interval or ratio data. The technique for computing the test statistics is straightforward but too lengthy to explain here. For the details, the interested reader may wish to consult [73]. The calculation technique is as follows:

Take into account that $(x_1, \ldots, x_m)$ and $(y_1, \ldots, y_n)$ are different pairs of separate sets of random variables. Also, let each set's arbitrary response be denoted by x and y. In addition, take into account that $y\sim$ G and $x\sim$ F, and let the data of observations be categorized. The function of Mann-Whitney is as follows:

$$\phi = h_{MW}(F,G) = Pr[x<y] + \frac{1}{2}Pr[x=y] \tag{18}$$

The $\widehat{G}$ and $\widehat{F}$ experimental distributions can be used to estimate the ϕ as shown below:

$$\widehat{\phi} = h_{MW}(\widehat{F},\widehat{G}) = \frac{1}{mn}(S_y - \frac{n(n+1)}{2}) \tag{19}$$

The mid-ranks are calculated by ranking all N $= m+n$ replies combined, breaking ties randomly, and average the tied values. Where S_y is regarded the total of the n mid-

Table 2 Results obtained by GWO-FC and FCNN in terms of MSE

Dataset	Method	MSE			
		Worst	Best	Average	Std
Albrecht	FCNN	3.12E−02	3.12E−02	3.12E−02	7.06E−18
	GWO-FC	1.84E−06	4.57E−07	1.06E−06	7.12E−07
China	FCNN	2.88E−03	2.88E−03	2.88E−03	2.21E−18
	GWO-FC	1.34E−03	1.34E−03	1.34E−03	0.00E+00
Cosmic	FCNN	9.79E−02	9.79E−02	9.79E−02	2.82E−17
	GWO-FC	2.57E−05	1.55E−08	1.02E−06	4.66E−06
COCOMO81	FCNN	3.19E−02	3.19E−02	3.19E−02	1.41E−17
	GWO-FC	1.88E−03	2.82E−04	1.03E−03	4.39E−04
COCOMONASA-I	FCNN	2.53E−02	2.53E−02	2.53E−02	1.76E−17
	GWO-FC	7.06E−03	4.58E−05	4.54E−03	3.36E−03
COCOMONASA-II	FCNN	2.01E−02	2.01E−02	2.01E−02	3.53E−18
	GWO-FC	1.17E−02	1.17E−02	1.17E−02	7.06E−18
Desharnais	FCNN	3.47E−02	3.47E−02	3.47E−02	7.06E−18
	GWO-FC	5.21E−03	1.85E−03	3.36E−03	8.44E−04
Kemerer	FCNN	5.87E−02	5.87E−02	5.87E−02	0.00E+00
	GWO-FC	1.72E−05	8.81E−08	1.85E−06	3.46E−06
Kitchenham	FCNN	3.38E−02	3.38E−02	3.38E−02	2.12E−17
	GWO-FC	1.71E−02	6.12E−03	1.05E−02	3.50E−03
Maxwell	FCNN	1.77E−02	1.77E−02	1.77E−02	0.00E+00
	GWO-FC	4.34E−03	3.15E−05	1.40E−03	1.81E−03
Miyazaki 94	FCNN	1.11E−02	1.11E−02	1.11E−02	5.29E−18
	GWO-FC	3.47E−03	3.47E−03	3.47E−03	1.32E−18
USP05	FCNN	1.21E−02	1.21E−02	1.21E−02	5.29E−18
	GWO-FC	4.73E−03	4.64E−03	4.73E−03	1.67E−05

ranks from the next group. Using the $\widehat{\phi}$, the Wilcoxon–Mann–Whitney Test is a permutation test.

Table 4 presents the outcomes of the Wilcoxon test according to the average of the best results obtained across 30 independent experiment runs. Regression and correlation values were used in this test to compare the two methods (GWO-FC and FCNN), as well as to assess how different the two estimation methods were from each other. The likelihood of the hypothesis's random validity is represented by the ρ-value. Confirmation versus the null hypothesis and high statistical significance are shown by a low ρ-value, because the null hypothesis is true. Depending on the ρ-value, the null hypothesis is either rejected or accepted. If the ρ-value is less than or equal to the probability threshold (α), which is the case, the null hypothesis is not supported.

Since the experimental factors affected the experiment's digital observations in this test, where $\alpha = 0.05$, any probability value less than α meant that less than 5% of the experiment results were due to chance and not to the experimental factors. As a result, there was a difference in the statistical significance between the two methods. From Table 4, it can be seen that the statistical indicators demonstrate that the results obtained by the GWO-FC were significantly different from those of the traditional FCNN in all datasets.

4.5.2 Convergence rate analysis

To evaluate the proposed method in depth, a convergence rate analysis was performed. In this work, the maximum number of iterations was 300. The values of the parameters (weights and biases) were changed at each iteration. The parameters had random values at the early iterations, but as the number of iterations increased, these values dropped and so the FCNN became stuck in the local optimum. Therefore, using the GWO helped the FCNN to escape from any local optima using the GWO's ability regarding exploration and exploitation, which accelerated and enhanced the convergence. The parameters produced by the GWO give the method the ability to explore the search space around the related best solutions.

The convergence curves for the average values of the overall results for the GWO-FC versus the FCNN for each dataset are shown in Fig. 7. It can be seen from the curves that the integration of the GWO algorithm with the FCNN

Table 3 Obtained results by GWO-FC and FCNN in terms of RAE, MAE, VAF, ED, MdMRE, RMSE, RRSE, R^2, MD, SA, △

Dataset	Method	RAE	MAE	VAF	ED	MdMRE	RMSE	RRSE	R^2	MD	SA	|△|
Albrecht	FCNN	0.3775	0.1180	79.0230	1.3688	0.0652	0.1767	0.5079	74.1995	7.0810	0.2954	1.3094
	GWO-FC	0.0021	0.0006	99.9996	0.0052	0.0011	0.0007	0.0019	99.9996	0.0385	0.7897	4.6520
China	FCNN	0.4051	0.0310	87.8659	0.6567	0.5708	0.0536	0.3768	85.8050	4.6433	0.6010	11.9641
	GWO-FC	0.2855	0.0218	93.3808	0.4485	0.3815	0.0366	0.2573	93.3802	3.2722	0.7604	15.0653
Cosmic	FCNN	0.7500	0.2656	81.3625	0.6258	0.2787	0.3129	0.7621	41.9168	1.0624	0.4286	2.8564
	GWO-FC	0.0003	0.0001	100.00	0.0002	0.0001	0.0001	0.0003	100.00	0.0004	0.5483	3.9629
COCOMO81	FCNN	1.0363	0.1518	58.5010	0.7789	0.1725	0.1787	0.7802	39.1241	2.8849	0.7193	6.7439
	GWO-FC	0.0889	0.0130	99.4646	0.0731	0.0131	0.0168	0.0733	99.4633	0.2476	0.9679	11.7360
COCOMONASA-I	FCNN	0.5455	0.1063	73.1959	0.6749	0.7871	0.1591	0.5528	69.4408	1.9136	0.4382	3.5230
	GWO-FC	0.0246	0.0048	99.9448	0.0287	0.0063	0.0068	0.0235	99.9447	0.0864	0.5348	4.8351
COCOMONASA-II	FCNN	0.5381	0.0967	63.7860	0.7505	0.8245	0.1418	0.6031	63.6300	2.7086	0.6381	5.1493
	GWO-FC	0.3632	0.0653	78.8259	0.5728	0.6308	0.1082	0.4603	78.8112	1.8284	0.6934	6.3965
Desharnais	FCNN	0.6609	0.1268	53.7076	0.9122	0.4391	0.1862	0.7416	45.0058	3.0430	0.4173	4.2956
	GWO-FC	0.1671	0.0321	97.0785	0.2106	0.0843	0.0430	0.1712	97.0697	0.7695	0.5301	6.0354
Kemerer	FCNN	0.5864	0.2021	59.7163	0.5418	0.2926	0.2423	0.6378	59.3243	1.0105	0.4581	1.8574
	GWO-FC	0.0008	0.0003	100.00	0.0007	0.0008	0.0003	0.0008	100.00	0.0014	0.5217	2.3965
Kitchenham	FCNN	0.6468	0.0981	37.5662	1.2200	0.4104	0.1839	0.8453	28.5507	4.3164	0.5380	3.4756
	GWO-FC	0.4055	0.0615	87.0676	0.5191	0.4622	0.0783	0.3596	87.0668	2.7059	0.6036	4.7639
Maxwell	FCNN	0.6902	0.1157	71.3400	0.5809	0.6550	0.1333	0.5665	67.9086	2.1982	0.4095	3.8723
	GWO-FC	0.0223	0.0037	99.9434	0.0245	0.0246	0.0056	0.0239	99.9430	0.0710	0.6740	5.8790
Miyazaki 94	FCNN	0.4909	0.0709	85.9358	0.4081	0.5017	0.1054	0.4345	81.1190	1.0632	0.4295	1.4053
	GWO-FC	0.3528	0.0509	94.4088	0.2281	0.5583	0.0589	0.2429	94.0987	0.7642	0.6492	2.5001
USP05	FCNN	1.5246	0.0762	37.8199	0.8506	3.3709	0.1098	0.8362	30.0812	4.5697	0.3905	4.8737
	GWO-FC	0.8443	0.0422	74.4427	0.5278	2.7289	0.0681	0.5189	73.0749	2.5307	0.4610	6.7600

significantly enhanced the convergence rate, as well as the accuracy of FCNN estimation, which in turn improved the quality of the results.

The findings of the experiments suggest that the usage of metaheuristic techniques in general, and the GWO in particular, may produce significant gains in the field of parameter optimization, since generating high quality results for ANNs is correlated to parameter optimization, which in turn enhances estimate effectiveness. Therefore, the use of metaheuristic methods to estimate optimal parameter values may enable ANNs to address the uncertainties in estimations for different benchmark datasets and provide more accurate results. In addition, the use of parameter optimization methods based on metaheuristic algorithms increases the chance of a neural network being able to enhance estimation accuracy and convergence speed, as well as reduce the possibility of falling into local optima.

4.5.3 Computational time

In order to compare the selected methods further, a computational time comparison between the traditional FCNN and the proposed GWO-FC in the simulation phase was performed using the test data. This comparison was made in Windows 10 64 bit, i5−10600 CPU@3.30 GHz, with 16 RAM. The results are presented in Table 5 for all datasets used by computing the mean time for 30 runs.

The results show that FCNN took less time compared to the GWO-FC method. The noticeable increase in the GWO-FC computation time is due to the fact that it consists of the FCNN training time plus the GWO optimization time. As a result, the GWO-FC method has several loops, as well as the process of passing data across the methods. Moreover, the calculated time that was considered here is for the total time of the FCNN training process and the GWO optimization process, where the bulk of the time is spent on the optimization process. Since the training and optimization processes are the same for all employed datasets, in addition to the fact that a small dataset (i.e., USP05) consumes more computational time than a large dataset (i.e., China), this is due to the complexity of the database itself. The same is true of the COCOMO81 and the COCOMONASA-II datasets, where COCOMONASA-II is comparatively larger than COCOMO81, but COCOMO81 takes more computational time than COCOMONASA-II.

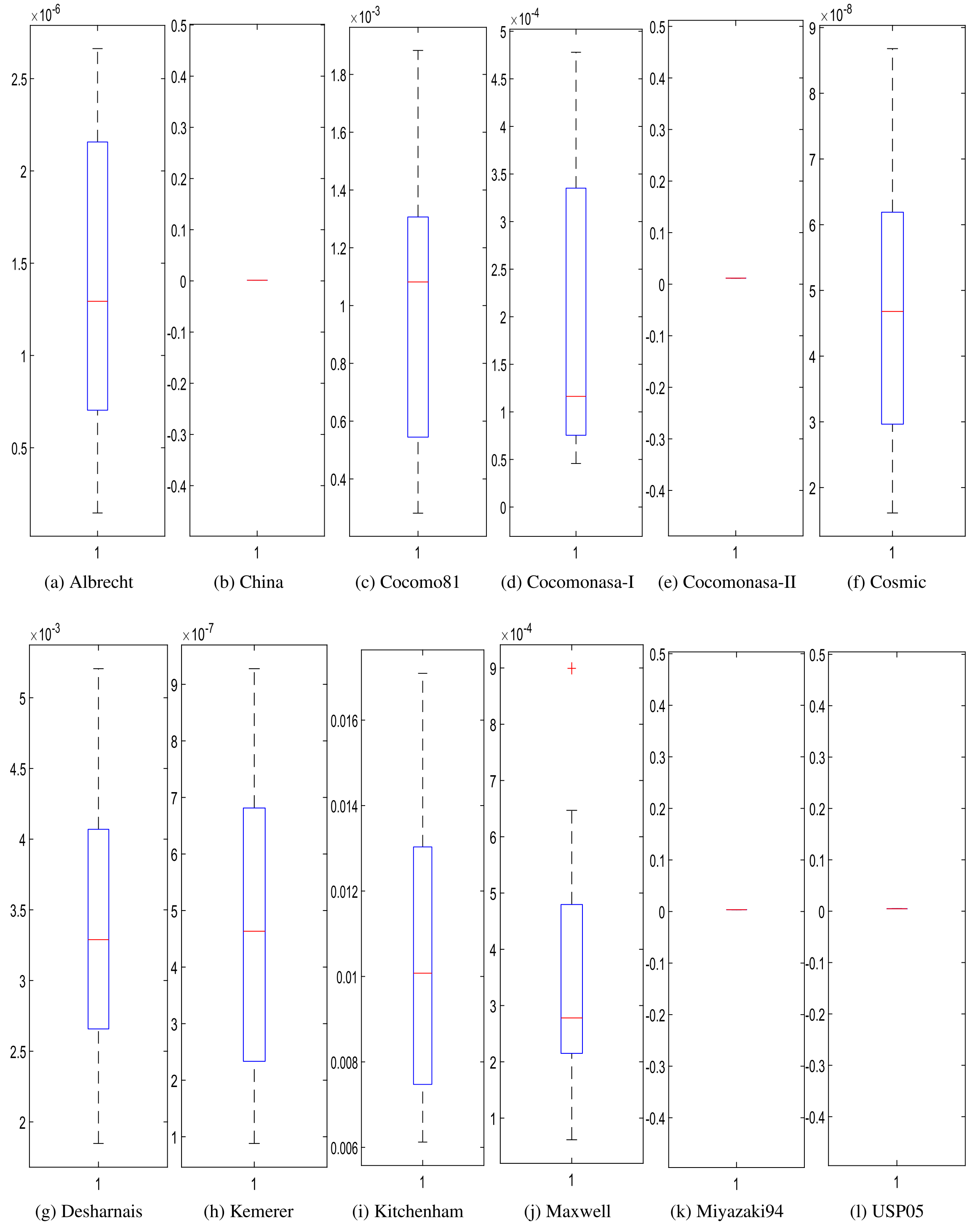


Fig. 6 Boxplots of GWO-FC

4.6 Validation comparison of GWO-FC against state-of-the-art methods

To support the proposed methodology, a comparison with state-of-the-art approaches from the literature was conducted. The comparison was performed based on experiments employing analogous evaluation criteria and benchmark datasets as in the previously described experiment. The results of the comparison are presented in five tables: Table 6 presents MSE and MAE, Tables 7, 8 and 9 present MAE, and Table 10 presents SA and (△).

Table 4 ρ-values of Wilcoxon Test for GWO-FC against FCNN

Dataset	ρ-values
Albrecht	5.96E−13
China	8.44E−15
COCOMO81	6.06E−13
COCOMONASA-I	2.31E−13
COCOMONASA-II	8.44E−15
Cosmic	6.06E−13
Desharnais	6.06E−13
Kemerer	6.06E−13
Kitchenham	6.06E−13
Maxwell	5.66E−13
Miyazaki 94	8.44E−15
USP05	1.35E−14

Table 5 Comparison of computational time

Dataset	FCNN	GWO-FC
Albrecht	0.3663	80.2685
China	0.4157	80.8358
COCOMO81	0.3729	80.3617
COCOMONASA-I	0.4522	79.9746
COCOMONASA-II	0.3767	79.7782
Cosmic	0.3659	80.4361
Desharnais	0.3717	80.8466
Kemerer	0.3975	80.3391
Kitchenham	0.3666	80.4654
Maxwell	0.3621	80.2819
Miyazaki 94	0.3883	80.4830
USP05	0.3857	81.9378

* Measurements in seconds

Table 6 shows that the GWO-FC was superior to Salp Swarm Algorithm with Backpropoagation Neural Network (SSA-BPNN) method in most of the datasets except in Cosmic and COCOMONASA-II in terms of MSE, in Miyazaki 94 in terms of MAE.

From Table 7, the proposed GWO-FC can be seen to be superior to other methods in most of the datasets except for China and Kitchenham, where SRF achieved the best results. The abbreviations used in Table 7 are as follow:

- Decision Tree (DT),
- Deep Net (DN),
- Elastic Net Regression (EN),
- Ensemble Technique: Bagging (BA),
- Ensemble Technique: Boosting (BS),
- Ensemble Technique: Weighted Averaging (WAVG),
- LASSO Regression (LASSO),
- Random Forest (RF),
- Ridge Regression (Ridge),
- Stacking Using RF (SRF)

Table 8 shows that the GWO-FC approach outperformed the competition over all datasets. The abbreviations available in Table 8 are as follow:

- Genetic algorithm - hybrid search-based algorithm (GA-HSBA)
- Black hole optimization algorithm - hybrid search-based algorithm (BHO-HSBA)
- Firefly algorithm (FFA) - hybrid search-based algorithm (FFA-HSBA)

Table 9 shows that the GWO-FC approach outperformed the competition over all datasets. The abbreviations available in Table 9 are as follow:

- Ant colony optimization (ACO)
- Chaos optimization algorithm (COA)
- Genetic algorithm (GA)
- Partial swarm optimization (PSO)
- Bat algorithm (BA)

Table 10 shows that the GWO-FC approach outperformed the competition over all datasets. The abbreviations available in Table 10 are as follow:

- Cluster-based fuzzy regression tree (CFRT)
- Multi-layer perceptron (MLP)
- K-nearest neighbor (KNN)
- Classification and regression trees (CART)
- Linear regression (LR)

In conclusion, the comparative results generally demonstrate that the proposed GWO-FC method can outperform other methods because it is robust and can handle a variety of situations that are different in complexity and dimension.

4.7 Statistical test analysis

To investigate the significance and variations among the outcomes of a proposed method and competitive methods, statistical analysis is crucial. A theoretically and empirically based analysis of potential statistical tests was applied to this research problem to compare two or more predictors/classifiers across multiple datasets included non-parametric tests (Wilcoxon and Friedman tests), parametric tests (paired ANOVA test). The non-parametric test assumes no commensurability of the results (sign test). In the theoretical part, we specifically addressed how a typical ML dataset can deviate from the basic assumptions of the

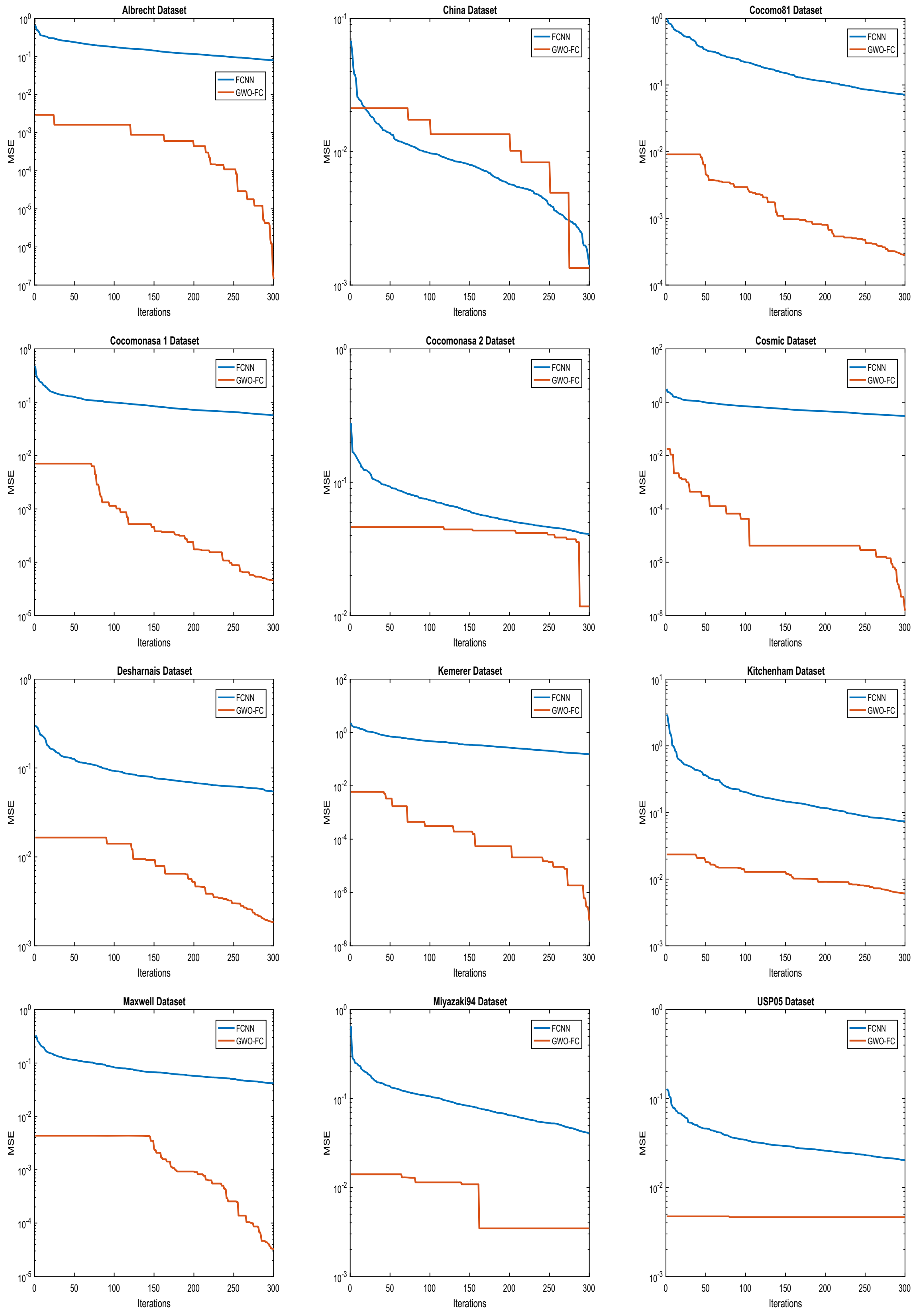


Fig. 7 Convergence curves of GWO-FC against FCNN

Table 6 Results obtained by GWO-FC against state-of-the-art methods [57]

Dataset	SSA-BPNN	GWO-FC	SSA-BPNN	GWO-FC
	MSE		MAE	
Albrecht	1.61E−02	**1.06E−06**	9.86E−02	**6.00E−04**
China	3.00E−03	**1.34E−03**	2.92E−02	**2.18E−02**
Cosmic	**1.34E−07**	1.02E−06	2.87E−02	**1.00E−04**
COCOMO81	1.60E−02	**1.03E−03**	1.01E−01	**1.30E−02**
COCOMONASA-I	7.40E−03	**4.54E−03**	7.02E−02	**4.80E−03**
COCOMONASA-II	**1.03E−02**	1.17E−02	6.59E−02	**6.53E−02**
Desharnais	1.57E−02	**3.36E−03**	9.78E−02	**3.21E−02**
Kemerer	1.62E−04	**1.85E−06**	1.17E−02	**3.00E−04**
Kitchenham	2.29E−02	**1.05E−02**	1.02E−01	**6.15E−02**
Maxwell	6.80E−03	**1.40E−03**	6.51E−02	**3.70E−03**
Miyazaki 94	3.50E−03	**3.47E−03**	**4.23E−02**	5.09E−02
USP05	1.44E−02	**4.73E−03**	6.05E−02	**4.22E−02**

*Best results in bold

Table 7 Results obtained by GWO-FC against state-of-the-art methods [74]

Dataset	Albrecht	China	Cosmic	COCOMO81	Kemerer	Kitchenham	Maxwell
	MAE						
RF	0.1940703	0.03832486	0.11619460	0.06047924	0.20769360	0.104712400	0.23302240
DT	0.2299442	0.02229970	0.10672270	0.08838886	0.37092710	0.117400800	0.28969550
Ridge	0.2495593	0.01977087	0.08810373	0.07912894	0.19713350	0.043428920	0.21629830
LASSO	0.2672776	0.01344521	0.08731722	0.08097285	0.18396700	0.038768060	0.21066170
EN	0.2522743	0.01406866	0.08874032	0.07980254	0.18900740	0.039611220	0.21127400
DN	0.2702398	0.09730134	0.19138450	0.25489060	0.40118950	0.140050100	0.29017840
WAVG	0.1658832	0.05308728	0.13789110	0.06493144	0.18110160	0.027151460	0.12118710
BA	0.1784421	0.01207605	0.11952850	0.08604002	0.20429140	0.009002502	0.23560230
BS	0.1237017	0.01059957	0.10725950	0.08095949	0.23457430	0.009002502	0.08203951
SRF	0.0288617	**0.00401619**	0.07027704	0.02278088	0.07030604	**0.005384577**	0.03566583
GWO-FC	**0.0006000**	0.02180000	**0.00010000**	**0.01300000**	**0.00030000**	0.061500000	**0.00370000**

*Best results in bold

Table 8 Results obtained by GWO-FC against state-of-the-art methods [30]

Dataset	Desharnais	COCOMONASA-I	COCOMONASA-II
	MAE		
GA-HSBA	4.000008	2.198879	5.209081
BHO-HSBA	4.001168	2.198906	5.200578
FFA-HSBA	4.000000	2.635393	5.256450
GWO-FC	**0.032062**	**0.004800**	**0.065299**

*Best results in bold

Table 9 Results obtained by GWO-FC against state-of-the-art methods [37]

Dataset Method	COCOMO 81	Maxwell
	MAE	
ACO	5.6100	4.5400
COA	4.0700	2.8600
GA	4.7800	3.2200
PSO	5.0600	3.7200
BA	4.7100	3.6400
GWO-FC	**0.0143**	**0.1083**

*Best results in bold

Table 10 Results obtained by GWO-FC against state-of-the-art methods [75]

Dataset / Method	Albrecht	China	COCOMO81	Desharnais	Kemerer	Miyazaki 94
	SA					
LR	0.5910	0.5480	0.4460	0.5040	0.1960	0.5040
MLP	0.2610	0.5860	0.7290	0.4080	0.4920	0.3710
CART	0.5040	0.7460	0.9570	0.4240	0.3820	0.5870
KNN	0.6550	0.5470	0.8910	0.4480	0.4630	0.4760
CFRT	0.7710	0.4620	0.8820	0.4370	0.4730	0.4350
GWO-FC	**0.7897**	**0.7604**	**0.9679**	**0.5301**	**0.5217**	**0.6492**
Method	|Δ|					
LR	2.9600	10.3880	4.4840	5.3710	0.7410	1.8300
MLP	1.2740	11.5890	7.6200	4.1490	1.9450	1.3900
CART	2.6370	14.1440	10.0610	4.4210	1.4520	2.1520
KNN	3.3270	10.6470	9.2910	4.8550	1.7420	1.6880
CFRT	4.0580	8.9300	9.0400	4.8500	1.8000	1.5530
GWO-FC	**4.6520**	**15.0653**	**11.7360**	**6.0354**	**2.3965**	**2.5001**

*Best results in bold

tests. We concluded that non-parametric tests should be favored over parametric ones based on the well-known statistical features of the tests and our understanding of the ML data [76]. In addition, for statistical comparisons of classifiers, we recommend a collection of straightforward non-parametric tests that are secure and reliable [76], such as the Friedman test with the appropriate post-hoc tests for comparison of more classifiers across different datasets and the Wilcoxon signed ranks test for comparison of two classifiers [76].

To determine if there were statistically significant variations between the GWO-FC's accuracy and that of the cutting-edge methods listed in Table 7, Friedman and Holm/Hochberg statistical tests [77, 78] were used in this work. The efficiency and appropriateness of the proposed strategy were also confirmed using statistical test methods.

4.7.1 Friedman's tests of GWO-FC and state-of-the-art methods in terms of MAE

A non-parametric statistical test called the Friedman's test was created by the economist, Milton Friedman. When measuring an ordinal dependent variable, this test is used to determine whether there are any variations between the sets (treatments). When the identical parameter has been evaluated in multiple circumstances on the same participants, Friedman's test is utilized for a one-way repetitive measurement analysis of variance by ranks. The following are the hypotheses for the comparison of recurrent assessments:

- H0: throughout repeated measures, the distributions are the same (there is no substantial variation across the tested sets);
- H1: throughout repeated measures, the distributions are different (there is a substantial variation across the tested sets).

In addition, this test looks at the values of rankings by column after ranking each row (or block) jointly. The test statistic that Friedman suggests [79] is as follows:

$$T = \frac{12}{mk(k+1)} \sum_{j=1}^{k} R_j^2 - 3m(k+1) \tag{20}$$

where, sets number is donated by k, subjects number is donated by m, sum of the ranks for the j^{th} set is donated by R_j.

Suppose significant differences are detected between groups (treatments) by Friedman's test (i.e., the null hypothesis of Friedman's test is rejected). In this case, Holm's procedure (Wright 1992) will be performed as a post-hoc method for multiple comparison tests to determine which groups (treatments) differ from the others (unplanned comparisons). Holm's procedure is one of the earliest usages of stepwise algorithms in simultaneous inference. This method is an improvement of the Bonferroni procedure [79] which applies a criterion's unequal allocation to each hypothesis being tested. A step-down method of Holm's tests the hypotheses in order of relevance in a sequential manner, and adjusts the crucial value in order to reject the null hypothesis. The more significant of the surviving hypotheses are successively taken into

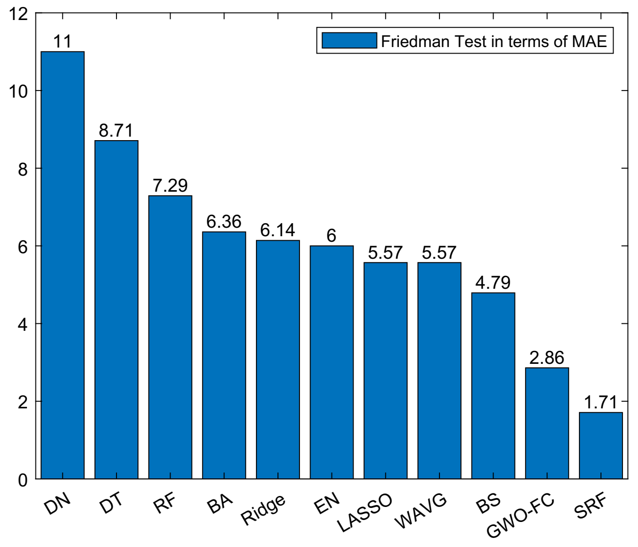


Fig. 8 Friedman's Test average rankings for comparative methods results in Table 7

consideration during the approach. Holm's test rejects the hypothesis linked to the most significant test statistic.

Friedman's test was performed as a quantitative evaluation of the statistical difference between the GWO-FC method and the state-of-the-art methods in terms of MAE, using a significance level of $\alpha = 0.05$. By using the results from Table 7 and Friedman's test, the GWO-FC was ranked against its rivals. Figure 8 provides an overview of how all of the competing approaches ranked.

Figure 8 shows that the GWO-FC method ranked second, with an average of 2.86, preceded by the SRF algorithm (average ranking 1.71), and followed by, in descending order, BS (4.78), WAVG (5.57), LASSO (5.57), EN (6.00), Ridge (6.14), BA (6.36), RF (7.29), DT (8.71), and DN (11.00). Also, the ρ-value calculated using Friedman's test for results listed in Table 7 was 1.1803E-5, which is below the threshold for significance ($\alpha = 0.05$). These results illustrate the extent to which there was a significant difference between the methods under evaluation.

4.7.2 Holm's/Hochberg statistical test analysis in terms of MAE

A post-hoc statistical method for contrasting the control method (i.e., the best performing method) with other methods is the Holm's/Hochberg test. This test was applied to the proposed and compared methods because there were significant statistical differences between the obtained results. In addition, the null hypothesis of equivalent accuracy was rejected by the Holm's/Hochberg method, with the aim of confirming the existence of significant differences in the accuracy of the results produced by the competing methods. The Holm's/Hochberg results are provided in Table 11. For all test instances, the confidence threshold level was 0.05.

Table 11 shows that the control method (SRF) was statistically better than the compared methods based on the results of the Holm's/Hochberg test. Additionally, the difference between the performance of the SRF and the GWO-FC was slight. The SRF and GWO-FC ρ-values based on the Holm's/Hochberg test were too close. Accordingly, the average estimation findings of the GWO-FC approach were statistically superior and significantly more compelling than those of the compared state-of-the-art approaches, except for the SRF method. Therefore, the GWO-FC method can be considered a viable alternative for estimating software effort and other engineering problems.

The superiority of the proposed method can be attributed to the synergy of estimation and optimization achieved by combining the GWO with an FCNN, enabling the GWO to explore the FCNN parameter space and discover the optimal subset of parameters values. This provides the best estimation performance and a reasonable balance between exploitation and exploration, as well as preventing the FCNN from falling into local optima.

Table 11 GWO-FC and other comparative methods Holm's/Hochberg results in Table 7

i	Method	ρ-value	Holm's/Hochberg	$\alpha \div i$	Null Hypotheses H_0
10	DN	0.0000	5.2378	0.0050	Rejected
9	DT	0.0001	3.9485	0.0056	Rejected
8	RF	0.0017	3.1427	0.0063	Rejected
7	BA	0.0088	2.6189	0.0071	Rejected
6	Ridge	0.0125	2.4981	0.0083	Rejected
5	EN	0.0156	2.4175	0.0100	Rejected
4	LASSO	0.0296	2.1757	0.0125	Rejected
3	WAVG	0.0296	2.1757	0.0167	Rejected
2	BS	0.0832	1.7325	0.0250	Rejected
1	GWO-FC	0.5191	0.6447	0.0500	Rejected

4.8 Discussion

This research demonstrates that combining a metaheuristic optimizer with an ML estimator, in this case, the GWO and the FCNN network, can improve the accuracy of the estimation process and achieve high quality estimation performance and results. Using the GWO to optimize the FCNN network significantly helps in optimizing the solution in an iterative search process. As a result, the GWO-FC method helped identify the optimal set of parameters for estimation activities. The results of the experiments demonstrated the value of parameter optimization, as a pre-processing step in any estimation process. As a result, there is a strong likelihood that including non-optimal parameter values degrades estimation quality. This is supported by the evaluation comparison results in Sect. 4.5 and the verification comparison results in Sect. 4.6. Furthermore, all the results show that the proposed integration approach was successful in identifying the best parameter values for the estimation process.

Another notable finding from this study is that the proposed GWO-FC performed significantly better than the traditional FCNN because the GWO maintains a balance among exploitation and exploration. Essentially, the GWO has significant exploration ability in the GWO-FC approach due to the embedded adaptive parameters [38]. Simultaneously, the FCNN contributes to the enhancement of local search capability, which improves exploitation. These are the main explanations for the results shown in Tables 2 and 3, which show that the GWO-FC outperformed the conventional FCNN not only in terms of MSE but also in a variety of other performance metrics.

In summary, the empirical findings show that combining GWO and FCNN is beneficial. In the GWO-FC, the GWO receives multiple new solutions from the FCNN during the training step, which maximizes the exploration capability. The GWO then evaluates the most promising solutions and selects the best one for use in the FCNN testing step (the estimation process). Based on the experimental results, it can be inferred that the FCNN's performance improved and became more stable, and that the optimal parameter values guarantee more accurate estimation. Finally, the proposed method can be said to be superior to comparable methods in the literature based on the statistical analyses.

In the future, researchers may wish to combine the GWO optimizer with another local-search algorithm, for instance, simulated annealing (SA), to improve search exploitation (local) capability, or to employ the GWO in other estimation/classification fields, such as medical diagnosis, intrusion detection, or image segmentation.

5 Threats to the study's validity

In the following subsections, a number of threats to the study's validity are covered:

5.1 Construct validity

If dependent and independent variables are not measured properly, a construct threat arises [80]. The datasets for the current research were taken from a trustworthy software engineering source and therefore this threat is not present here.

5.2 Internal validity

When a study's usage of software metrics is loosely connected to the software efforts, threats to internal validity become possible. Internal validity threats are possible in this study as the programmer's expertise capacity was not considered.

5.3 External validity

The study's conclusions may be generalized since several project kinds were represented in the datasets and so there is very little threat to the study's external validity.

6 Conclusion

Parameter tuning is a difficult optimization challenge for engineering problems involving estimation and classification because the aim of such tuning is to maximize and strengthen the performance of the estimator used. Previous studies have shown that metaheuristic techniques are appropriate for addressing this issue. Therefore, in this study, a promising metaheuristic algorithm, the gray wolf optimizer (GWO), was combined with the fully connected neural network (FCNN) method, named (GWO-FC), for software development effort estimation (SEE) problems. In GWO-FC, the GWO is utilized to optimize the FCNN parameters (weights and biases) to increase the accuracy of FCNN estimation by defining the most suitable parameter values to tackle the SEE problem. Hence, the GWO helps to increase exploration ability in the parameter search field as well as prevent the FCNN from falling into local optima.

The proposed GWO-FC method was evaluated against the traditional FCNN. In addition, the proposed method was validated against 24 state-of-the-art methods extracted from the literature. The research findings demonstrate that, for the majority of benchmark datasets and evaluation criteria, the GWO-FC substantially improved on the FCNN and most recent approaches. The results indicate that the traditional FCNN has limitations when trying to address the estimation problem. On the other hand, the GWO-FC has the potential to tackle the estimation task by increasing exploration in the search space. Thus, the results demonstrate that the GWO can be integrated with ML methods for the purpose of maximizing the accuracy of the estimation task.

Nevertheless, it should be stated that the proposed method still lacks the ability to compete with other modified metaheuristic algorithms in the literature. In addition, the proposed method suffers from a relatively high computational time compared to the traditional FCNN. Therefore, further studies may wish to consider developing new approaches to address the estimation problem. The authors intend to develop new and more efficient methods of tackling the SEE problem in the future. Also, a more efficient method for computational time can be developed by which the proposed method is improved.

Since the GWO algorithm's limitations, in some circumstances, may prolong convergence time, plans have been made to develop novel techniques to improve the search behavior of the algorithm so that it maintains an appropriate balance between exploitation and exploration, while relatively increasing the exploration capacity. Metaheuristic techniques have been effectively used in several fields of study to enhance the training of ANNs but there are still few pertinent papers in this field. Future research should focus on investigating the use of metaheuristic algorithms in ANN architectures for SEE.

Author contributions SK suggested, represented, and coded the contributions with support from MA. SK accomplished the experimental design, and he executed the program and experimental scenarios on the PC with support from MA. Both SK and MA develop the mathematical models for the proposed approach. SK supervised and led the project as well as distributed and assigned the tasks to the authors of the paper. As a result, Kassaymeh deserves to be the first author. MA and MAB helped in designing the experiments and they wrote the main parts of the paper with support from AIH and SK. MA also assisted in drawing the flowcharts and writing the pseudo-code for the algorithm. MAB and AIH validated the developed code through several tests. AIH managed the results and tables. He collected and summarized the results and critiqued the comparative methods. AIH did the interpretation of the results with support from MAB and SK. He also performed the Wilcoxon statistical test. Furthermore, he discussed the results of the statistical test. MAM summarizes the literature, collects the results from the PC, and prepares the excel sheet. AIH and MAM validate the dataset reading and processing of the proposed approaches. They also check the technical concepts in the paper, ensure its readability, and support the English proof. They also support checking the references. Hammouri and Al-Ma'aitah provided critical feedback and helped shape the research, analysis, and manuscript.

Funding The research reported in this publication was supported by the Deanship of Scientific Research and Innovation at Al-Balqa Applied University in Jordan, Grant Number: DSR-2020~#295.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

References

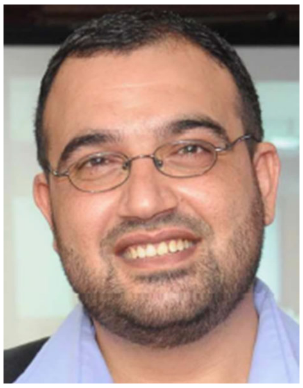
1. Idri, A., Hosni, M., Abran, A.: Systematic literature review of ensemble effort estimation. J. Syst. Softw. **118**, 151–175 (2016)
2. Gautam, S.S., Singh, V.: The state-of-the-art in software development effort estimation. J. Softw.: Evol. Process **30**(12), e1983 (2018)
3. Karimi, A., Gandomani, T. J.: Software development effort estimation modeling using a combination of fuzzy-neural network and differential evolution algorithm. Int. J. Electr. Comput. Eng. **11**(1), 2088–8708
4. Nassif, A. B., Azzeh M., Idri, A., Abran A.: Software development effort estimation using regression fuzzy models, Comput Intell. Neurosci. **2019**, 8367214
5. Abdelali, Z., Hicham, M., Abdelwahed, N.: An ensemble of optimal trees for software development effort estimation. In: International Conference on Advanced Information Technology, Services and Systems, pp. 55–68. Springer (2018)
6. Eduardo Carbonera, C., Farias, K., Bischoff V.: Software development effort estimation: a systematic mapping study. IET Softw. **14**(4), 328–344 (2020)
7. Wen, J., Li, S., Lin, Z., Hu, Y., Huang, C.: Systematic literature review of machine learning based software development effort estimation models. Info. Softw. Technol. **54**(1), 41–59 (2012)
8. Kaushik A., Choudhary N., et al.: Software cost estimation using lstm-rnn. In: Proceedings of International Conference on Artificial Intelligence and Applications, pp. 15–24. Springer (2021)
9. Fadhil, A.A., Alsarraj, R.G., Altaie, A.M.: Software cost estimation based on dolphin algorithm. IEEE Access **8**, 75279–75287 (2020)
10. Ghatasheh, N., Faris, H., Aljarah, I., Al-Sayyed, R. M.: Optimizing software effort estimation models using firefly algorithm. Comput. Sci. **8**(3), 133–142 (2019)
11. A. Idri, F. azzahra Amazal, A. Abran, Analogy-based software development effort estimation: a systematic mapping and review, Info. Softw. Technol. **58**, 206–230 (2015)
12. Rankovic, N., Rankovic, D., Ivanovic, M., Lazic, L.: A new approach to software effort estimation using different artificial neural network architectures and taguchi orthogonal arrays. IEEE Access **9**, 26926–26936 (2021)
13. Mahmood, Y., Kama, N., Azmi, A., Khan, A. S., Ali, M.: Software effort estimation accuracy prediction of machine learning techniques: a systematic performance evaluation. Softw. Practice Exp. **52**(1), 39–65 (2022)
14. Albashish, D., Al-Sayyed, R., Abdullah, A., Ryalat, M. H., Almansour N. A.: Deep cnn model based on vgg16 for breast cancer classification. In: 2021 International Conference on Information Technology (ICIT), pp. 805–810. IEEE (2021)
15. Rahman, M. A., Chandren Muniyandi, R., Albashish, D., Rahman, M. M., Usman, O. L.: Artificial neural network with taguchi

method for robust classification model to improve classification accuracy of breast cancer. PeerJ Comput. Sci. **7**, e344 (2021)
16. Ali, A., Gravino, C.: A systematic literature review of software effort prediction using machine learning methods. J. Softw. Evol. Process **31**(10), e2211 (2019)
17. Nanassif, A. B., Azzeh, M., Capretz, L. F., Ho, D.: Neural network models for software development effort estimation: a comparative study. Neural Comput. Appl. **27**(8), 2369–2381 (2016)
18. Nassif, A.B., Ho, D., Capretz, L.F.: Towards an early software estimation using log-linear regression and a multilayer perceptron model. J. Syst. Softw. **86**(1), 144–160 (2013)
19. Lopez-Martin, C.: Applying a general regression neural network for predicting development effort of short-scale programs. Neural Comput. Appl. **20**(3), 389–401 (2011)
20. López-Martín, C.: Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects. Appl. Soft Comput. **27**, 434–449 (2015)
21. Nassif, A. B., Capretz, L. F., Ho, D.: Software effort estimation in the early stages of the software life cycle using a cascade correlation neural network model. In: 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 589–594. IEEE (2012)
22. Shukla, S., Kumar, S.: Applicability of neural network based models for software effort estimation. In: IEEE World Congress on Services (SERVICES), Vol. 2642, pp. 339–342. IEEE (2019)
23. Mahmood, Y., Kama, N., Azmi, A., Ali, M.: Improving estimation accuracy prediction of software development effort: a proposed ensemble model. In: 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), pp. 1–6. IEEE (2020)
24. Hammouri, A.I., Braik, M.S., Al-Betar, M.A., Awadallah, M.A.: Isa: a hybridization between iterated local search and simulated annealing for multiple-runway aircraft landing problem. Neural Comput. Appl. **32**(15), 11745–11765 (2020)
25. Al-Betar, M.A., Alyasseri, Z.A.A., Awadallah, M.A., Doush, I.A.: Coronavirus herd immunity optimizer (chio). Neural Comput. Appl. **33**(10), 5011–5042 (2021)
26. Wang, L., Wu, B., Zhu, Q., Zeng, Y.-R.: Forecasting monthly tourism demand using enhanced backpropagation neural network. Neural Processing Letters **52**(3), 2607–2636 (2020)
27. Sun, W., Huang, C.: A carbon price prediction model based on secondary decomposition algorithm and optimized back propagation neural network. J. Clean. Prod. **243**, 118671 (2020)
28. Jiang, J., Chen, Z., Wang, Y., Peng, T., Zhu, S., Shi, L.: Parameter estimation for pmsm based on a back propagation neural network optimized by chaotic artificial fish swarm algorithm. Int. J. Comput. Commun. Control **14**(6), 615–632 (2020)
29. Shen, X., Zheng, Y., Zhang, R.: A hybrid forecasting model for the velocity of hybrid robotic fish based on back-propagation neural network with genetic algorithm optimization. IEEE Access **8**, 111731–111741 (2020)
30. Rhmann, W., Pandey, B., Ansari, G.A.: Software effort estimation using ensemble of hybrid search-based algorithms based on metaheuristic algorithms. Innov. Syst. Softw. Eng. **18**(2), 309–319 (2022)
31. Ardiansyah, A., Ferdiana, R., Permanasari, A.E.: Mucpso: a modified chaotic particle swarm optimization with uniform initialization for optimizing software effort estimation. Appl. Sci. **12**(3), 1081 (2022)
32. Khuat, T.T., Le, M.H.: A novel hybrid abc-pso algorithm for effort estimation of software projects using agile methodologies. J. Intell. Syst. **27**(3), 489–506 (2018)
33. Parizi, M.K., Keynia, F., Bardsiri, A.K.: Hscwma: a new hybrid sca-wma algorithm for solving optimization problems. Int. J. Inf. Technol. Decis. Making **20**(02), 775–808 (2021)
34. Ullah, A., Wang, B., Sheng, J., Long, J., Asim, M., Sun, Z.: Optimization of software cost estimation model based on biogeography-based optimization algorithm. Intell. Decis. Technol. **14**(4), 441–448 (2020)
35. Resmi, V., Vijayalakshmi, S., Chandrabose, R.S.: An effective software project effort estimation system using optimal firefly algorithm. Clust. Comput. **22**(5), 11329–11338 (2019)
36. Arora, M., Verma, S., Wozniak, M., Shafi, J., Ijaz, M.F., et al.: An efficient anfis-eebat approach to estimate effort of scrum projects. Sci. Rep. **12**(1), 1–14 (2022)
37. Khan, M.S., Jabeen, F., Ghouzali, S., Rehman, Z., Naz, S., Abdul, W.: Metaheuristic algorithms in optimizing deep neural network model for software effort estimation. IEEE Access **9**, 60309–60327 (2021)
38. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
39. Al-Betar, M.A., Awadallah, M.A., Krishan, M.M.: A non-convex economic load dispatch problem with valve loading effect using a hybrid grey wolf optimizer. Neural Comput. Appl. **32**(16), 12127–12154 (2020)
40. Chen, X., Yi, Z., Zhou, Y., Guo, P., Farkoush, S.G., Niroumandi, H.: Artificial neural network modeling and optimization of the solid oxide fuel cell parameters using grey wolf optimizer. Energy Rep. **7**, 3449–3459 (2021)
41. ul Hassan, C. A., Khan, M. S.: An effective nature inspired approach for the estimation of software development cost. In: 2021 16th International Conference on Emerging Technologies (ICET), pp. 1–6. IEEE (2021)
42. Emary, E., Zawbaa, H.M., Grosan, C.: Experienced gray wolf optimization through reinforcement learning and neural networks. IEEE Trans. Neural Netw. Learn. Syst. **29**(3), 681–694 (2017)
43. Sheta, A.F., Rine, D., Kassaymeh, S.: Software effort and function points estimation models based radial basis function and feedforward artificial neural networks. Int. J. Next-Generation Comput. **6**(3), 192–205 (2015)
44. Sheta, A.F., Kassaymeh, S., Rine, D.: Estimating the number of test workers necessary for a software testing process using artificial neural networks. IJACSA **5**(7), 186–192 (2014)
45. Agahian, S., Akan, T., Battle royale optimizer for training multilayer perceptron. Evol. Syst. **2021**, 1–13 (2021)
46. Kumar, P. S., Behera, H.: Role of soft computing techniques in software effort estimation: an analytical study. In: Computational Intelligence in Pattern Recognition, pp. 807–831. Springer (2020)
47. Jorgensen, M., Shepperd, M.: A systematic review of software development cost estimation studies. IEEE Trans. Soft. Eng. **33**(1), 33–53 (2006)
48. Heemstra, F.J.: Software cost estimation. Info. Softw. Technol. **34**(10), 627–639 (1992)
49. Azzeh, M., Nassif, A.B., Banitaan, S.: Comparative analysis of soft computing techniques for predicting software effort based use case points. IET Softw. **12**(1), 19–29 (2017)
50. Charette, R.N.: Why software fails [software failure]. IEEE Spectrum **42**(9), 42–49 (2005)
51. Gharehchopogh, F.S., Maleki, I., Khaze, S.R.: A novel particle swarm optimization approach for software effort estimation. Int. J. Acad. Res. **6**(2), 69–76 (2014)
52. Wang, Y., Wang, L., Chang, Q., Yang, C.: Effects of direct input-output connections on multilayer perceptron neural networks for time series prediction. Soft Comput. **24**(7), 4729–4738 (2020)
53. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989)

54. Ding, S., Su, C., Yu, J.: An optimizing bp neural network algorithm based on genetic algorithm. Artif. Intell. Rev. **36**(2), 153–162 (2011)
55. Han J., Pei, J., Kamber, M.: Data mining: concepts and techniques. Elsevier (2011)
56. Kassaymeh, S., Abdullah, S., Al-Betar, M. A., Alweshah, M.: Salp swarm optimizer for modeling the software fault prediction problem. J. King Saud Univ. Comput. Info. Sci. **34**, 3365 (2022)
57. Kassaymeh, S., Abdullah, S., Al-Laham, M., Alah, M., Al-Betar, M. A., Othman, Z.: Salp swarm optimizer for modeling software reliability prediction problems. Neural Process. Lett. **2021**, 1–37 (2021)
58. Heryanto, A., Gunanta, A.: High availability in server clusters by using backpropagation neural network method. J. Teknol. Open Sour. **4**(1), 08–18 (2021)
59. Luo, X., Shang, M., Li, S.: Efficient extraction of non-negative latent factors from high-dimensional and sparse matrices in industrial applications. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 311–319. IEEE (2016)
60. Montana, D. J., Davis, L., et al.: Training feedforward neural networks using genetic algorithms. In: IJCAI, Vol. 89, pp. 762–767. (1989)
61. Fong S., Deb, S., Yang X. S.: How meta-heuristic algorithms contribute to deep learning in the hype of big data analytics. In: Progress in intelligent computing techniques: theory, practice, and applications, pp. 3–25. Springer (2018)
62. Talbi, E. G.: Metaheuristics: from design to implementation. Wiley (2009)
63. Muthukumar, V., Narang, A., Subramanian, V., Belkin, M., Hsu, D., Sahai, A.: Classification vs regression in overparameterized regimes: does the loss function matter? J. Machine Learn. Res. **22**(222), 1–69 (2021)
64. Dornaika, F., Bekhouche, S.E., Arganda-Carreras, I.: Robust regression with deep cnns for facial age estimation: an empirical study. Exp. Syst. Appl. **141**, 112942 (2020)
65. Chen, X., Yu, R., Ullah, S., Wu, D., Li, Z., Li, Q., Qi, H., Liu, J., Liu, M., Zhang, Y.: A novel loss function of deep learning in wind speed forecasting. Energy **238**, 121808 (2022)
66. Keung, J., Kocaguneli, E., Menzies, T.: Finding conclusion stability for selecting the best effort predictor in software effort estimation. Autom. Softw. Eng. **20**(4), 543–567 (2013)
67. Albrecht, A. J., Gaffney, J. E.: Software function, source lines of code, and development effort prediction: a software science validation. IEEE Trans. Softw. Eng. **6**, 639–648 (1983)
68. Qi, F., Jing, X.-Y., Zhu, X., Xie, X., Xu, B., Ying, S.: Software effort estimation based on open source projects: case study of github. Info. Softw. Technol. **92**, 145–157 (2017)
69. Desharnais J.: Analyse statistique de la productivitie des projects informatique a partie de la technique des point des function. Masters Thesis University of Montreal (1989)
70. Kitchenham, B., Pfleeger, S.L., McColl, B., Eagan, S.: An empirical study of maintenance and development estimation accuracy. J. Syst. Softw. **64**(1), 57–77 (2002)
71. Tawosi, V., Sarro, F., Petrozziello, A., Harman, M.: Multi-objective software effort estimation: a replication study. IEEE Trans. on Softw. Eng. **48**,1–3 (2021)
72. Ali, A., Gravino, C.: Improving software effort estimation using bio-inspired algorithms to select relevant features: an empirical study. Sci. Comput. Program. **205**, 102621 (2021)
73. Bland, M.: An introduction to medical statistics. Oxford University Press, UK (2015)
74. Ag, P.V., Varadarajan, V., et al.: Estimating software development efforts using a random forest-based stacked ensemble approach. Electronics **10**(10), 1195 (2021)
75. Assia Najm A. M., Abdelali Z.: Cluster-based fuzzy regression trees for software cost prediction. Indonesian J. Electr. Eng. Comput. Sci. **27**(2), 1138–1150 (2022)
76. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Machine Learn. Res. **7**, 1–30 (2006)
77. Wang, Z., Li, M., Li, J.: A multi-objective evolutionary algorithm for feature selection based on mutual information with a new redundancy measure. Info. Sci. **307**, 73–88 (2015)
78. Canul-Reich, J., Hall, L.O., Goldgof, D.B., Korecki, J.N., Eschrich, S.: Iterative feature perturbation as a gene selector for microarray data. Int. J. Pattern Recognit. Artif. Intell. **26**(05), 1260003 (2012)
79. Sidney, S.: Nonparametric statistics for the behavioral sciences. J. Nervous Mental Dis. **125**(3), 497 (1957)
80. Zhou, Y., Leung, H., Xu, B.: Examining the potentially confounding effect of class size on the associations between object-oriented metrics and change-proneness. IEEE Trans. Softw. Eng. **35**(5), 607–623 (2009)

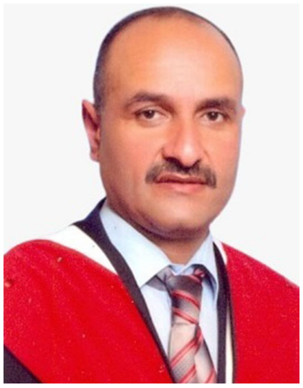
SOFIAN KASSAYMEH holds a master's degree in information systems-software engineering from the University of Hasselt, Belgium, in 2009 and a Ph.D. degree in Computer Science (Software Engineering and Artificial Intelligence) from UKM, Malaysia, in 2021. His research interests include combinatorial optimization, machine learning, swarm intelligence, and metaheuristic algorithms.

MOHAMMED ALWESHAH received his BS in computer sciences in 1993, from Al-Mustansiriah University, Iraq and his master degree of computer sciences in 2005, from Al-Balqa Applied University, Jordan. Mohammed Alweshah received his Ph.D. degree from the Computer Science Department, School of Information Technology, National University of Malaysia (UKM), Malaysia in 2013. He was working under the supervision of Prof. Salwani Abdullah. Currently, Alweshah is a faculty member with the Prince Abullah Bin Ghazi Faculty of Information Technology, Al- Balqa Applied University, AlSalt-Jordan since 2005. His research interests include Heuristics and Meta-heuristics, Evolutionary Algorithms, and data mining problems.



MOHAMMED AZMI AL-BETAR was at the Department of Computer Science, Al-Balqa Applied University, Irbid, Jordan. He is currently an Associate Professor with the Master of Articial Intelligence (MSAI) Program, College of Engineering and IT, Ajman University, the Director of the Articial Intelligence Research Center (AIRC), and the Head of the Evolutionary Computation Research Group. He has published more than 175 scientic publications in high quality and well-reputed journals and conferences. He ranked in Stanford study of the world's top 2% of scientists. His research interests include metaheuristic optimization methods and hard combinatorial optimization problems, including scheduling, engineering, and timetabling.

ABDELAZIZ I. HAMMOURI obtained his BSc in Information Technology from Al-Balqa Applied University, Jordan, and his MSc degree specializing in Computer Science from Al-Balqa Applied University. He did his Ph.D. in Computer Science at the National University of Malaysia (UKM). Now, he is an Associate Professor at the Department of Computer Science, Al-Balqa Applied University, Jordan. His research interest falls under Artificial Intelligence and Operation Research, particularly in meta-heuristic algorithms in optimization areas that involve different real-world applications and optimization problems, such as university timetabling, job shop scheduling, patient admission problems, space allocation, data clustering, feature selection, and classification problems.



MOHAMMAD ATWAH AL-MA'AITAH holds a master's degree in Information System, 2002 and a Ph.D. in Electronic Collaboration Systems in 2008. His research interests include information systems, e-business, cloud ERP, databases, and knowledge management.