

Received December 1, 2021, accepted December 8, 2021, date of publication December 13, 2021, date of current version December 31, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3135201

Artificial Neural Networks Hidden Unit and Weight Connection Optimization by Quasi-Refraction-Based Learning Artificial Bee Colony Algorithm

NEBOJSA BACANIN¹, TIMEA BEZDAN¹, K. VENKATACHALAM², MIODRAG ZIVKOVIC¹,
IVANA STRUMBERGER¹, MOHAMED ABOUHAWWASH^{3,4}, AND ABEER B. AHMED⁵

¹Department of Computer Science, Singidunum University, 11000 Belgrade, Serbia

²Department of Applied Cybernetics, Faculty of Science, University of Hradec Králové, 500 03 Hradec Králové, Czech Republic

³Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt

⁴Department of Computational Mathematics, Science, and Engineering (CMSE), Michigan State University, East Lansing, MI 48824, USA

⁵Software Engineering Department, College of Computing and Information Technology, Arab Academy for Science, Technology and Maritime, Cairo 12577, Egypt

Corresponding author: Nebojsa Bacanin (nbacanin@singidunum.ac.rs)

ABSTRACT Artificial neural networks are one of the most commonly used methods in machine learning. Performance of network highly depends on the learning method. Traditional learning algorithms are prone to be trapped in local optima and have slow convergence. At the other hand, nature-inspired optimization algorithms are proven to be very efficient in complex optimization problems solving due to derivative-free solutions. Addressing issues of traditional learning algorithms, in this study, an enhanced version of artificial bee colony nature-inspired metaheuristics is proposed to optimize connection weights and hidden units of artificial neural networks. Proposed improved method incorporates quasi-reflection-based learning and guided best solution bounded mechanisms in the original approach and manages to conquer its deficiencies. First, the method is tested on a recent challenging CEC 2017 benchmark function set, then applied for training artificial neural network on five well-known medical benchmark datasets. Further, devised algorithm is compared to other metaheuristics-based methods. The efficiency is measured by five metrics - accuracy, specificity, sensitivity, geometric mean, and area under the curve. Simulation results prove that the proposed algorithm outperforms other metaheuristics in terms of accuracy and convergence speed. The improvement of the accuracy over the other methods on different datasets are between 0.03% and 12.94%. The quasi-reflection-based learning mechanism significantly improves the convergence speed of the original artificial bee colony algorithm and together with the guided best solution bounded, the exploitation capability is enhanced, which results in significantly better accuracy.

INDEX TERMS Artificial neural network, optimization, metaheuristics, quasi-reflection-based learning, artificial bee colony.

I. INTRODUCTION

Machine learning can be defined as the application of artificial intelligence (AI) to enable computers to learn automatically, with the ability to improve from the previous experience. Therefore, machine learning is typically based on developing programs that are able to process data and learn from it. The main goal of machine learning is to enable computers and systems to learn on their own, without human interference.

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

Deep learning is a subdomain of machine learning, focused on the algorithms inspired by the human brain functionality and structure, that are known under the name artificial neural networks (ANNs). The ANNs are formed by a number of interconnected processing nodes (neurons) that transform a collection of inputs into a collection of outputs. The transformation is defined by the characteristics of the nodes, together with the weights of the connections between the nodes. It is possible to adapt the network by modification of the connections between neurons. During the learning process in ANNs, the weights and the biases are adjusted, and their values

makes stronger the connection between neurons in various layers. The value of the weights and biases are updated during the learning process, in a way to reduce the classification error rate, which is measured by loss function. The connection weight and bias value adjustment is also called neural network training. Neural network training belongs to the type of supervised learning, which learns from the labeled data, comparing the actual class to the predicted class.

There are two great challenges that neural networks facing. One is the network training, and another is, finding the appropriate network structure. The standard optimizers are typically used to train the neural networks. The survey of the available literature shows that metaheuristics can be successfully utilized instead of optimizers. The second challenge is to find the appropriate network structure for the given task, which is also known as the process of hyperparameter optimization. Both challenges are considered to be NP-hard problems by nature, in other words, they cannot be solved by traditional approaches in an acceptable amount of time. Instead, they require the application of stochastic approximation approaches, such as nature-inspired metaheuristics.

Deterministic and stochastic approaches are used for training ANNs. Gradient-based training and back-propagation [1] are used most commonly for neural network optimization, which are deterministic approaches, and they have a disadvantage of local optima stagnation, vanishing gradient and slow convergence. In the back-propagation (BP) methods, additional learning parameters should be determined, such as learning rate, momentum.

These issues motivated researchers to find algorithms and approaches which will avoid getting trapped in the local minima and to speed up the convergence. Addressing this issue, different derivative-free algorithms, such as metaheuristic algorithms have been applied for neural network training. First, in 1989, genetic algorithm (GA) is proposed by Montana and Davis [2] to train ANNs. In the paper, the results shows that GA outperforms BP on sonar image classification problems. Later, other metaheuristics are applied successfully for weight and bias optimization [3]–[6]. The utilization of metaheuristic based algorithms improves the search ability of neural network training for such values of weights and biases which will reduce the classification error rate more than gradient-based approaches.

In this study, an enhanced version of a well-known and widely applied artificial bee colony (ABC) swarm intelligence metaheuristics, that overcomes observed cons of the basic approach, is devised. Proposed improved method incorporates quasi-reflection-based learning and guided best solution bounded mechanisms in the basic artificial bee colony, and according to experimental findings manages to significantly improve convergence speed and results' quality of the original algorithm. Devised method was first tested on recent challenging CEC 2017 benchmark function set, then adapted and applied for training artificial neural network and evaluated on five well-known medical benchmark datasets.

This paper is motivated by the following research questions: How to develop a neural network training method to achieve higher accuracy and to speed up the training process? How to develop an efficient metaheuristic-based algorithm for neural network training? The objective of this work addresses these research questions as follows:

- develop improved ABC metaheuristic which outperforms the basic ABC and its variants in terms of convergence speed and quality of solutions;
- adopt the newly developed ABC method to optimize the connection weight and biases in the neural network, which results in better accuracy and faster execution than other existing methods and
- extend the experiment by hidden unit optimization, by keeping high accuracy and reducing the computation time.

The rest of this paper is organized as follows: Section II presents the background and related work, Section III provides an overview of artificial neural networks and their optimization, Section IV describes the original ABC algorithm, its deficiencies and the proposed method, Section V presents the CEC 2017 simulations, following by the experiment of artificial neural network training optimization and hidden unit optimization, and Section VI concludes the paper.

II. BACKGROUND AND RELATED WORK

There are numerous implementations of swarm intelligence metaheuristics, either in original or in enhanced/hybridized forms, that were tested against standard unconstrained and constrained benchmark functions set. Additionally, a large number of algorithms were validated on practical NP-hard challenges in various domains. One of the first algorithms from this group was particle swarm optimization (PSO), described in [7]. PSO mimics the behavior exhibited by the flocks of birds or fish, and it was used to solve different practical problems, such as the task scheduling problem in the cloud computing [8], [9]. Another famous swarm algorithm is the ant colony optimization (ACO) algorithm [10], which was inspired by the social behavior of the colony of ants. Artificial bee colony (ABC) is another well-known representative of swarm intelligence, which is considered to be a very efficient optimizer [11]. ABC has been tested against benchmark [12] and applied in solving various practical problems from different domains [13], [14]. Other well-known algorithms that belong to the group of swarm metaheuristics include the firefly algorithm (FA) [15], [16], the bat algorithm (BA) [17], [18], the whale optimization algorithm (WOA) [19], [20], and the elephant herding optimization (EHO) [21], [22]. Some of the newer swarm approaches include the moth search algorithm (MS), proposed by Wang in 2016 [23], which is considered to be one of the most efficient algorithms according to the test results against standard benchmark problems [24], and it had shown very promising results when it was applied in the real-world NP-hard scenarios, such as drone placement problem [25] and lifetime optimization in wireless sensor networks [16]. There is a great

number of domains and practical problems where swarm intelligence algorithms can be successfully applied. In some cases, swarm intelligence algorithms were able to achieve state-of-the-art results, including: path planning [26], node localization problem and energy efficiency in wireless sensor networks [27], [28], cloud computing and task scheduling [18], [29], [30], COVID-19 cases prediction [31], [32], feature selection problem [33], [34], ANN and CNN training optimization [35]–[39], text document clustering [40], as well as many others [41], [42].

The ANNs have a large domain of applications, ranging from the image classification task [35]–[38], time series prediction [31], [43], [44], to the wind speed forecasting [45]–[47]. The learning process of ANN is considered to be one of the most difficult challenges in machine learning and has attracted many researchers recently. Metaheuristics approaches have been widely used in the process of training artificial neural networks, as can be seen from the recent literature. Grasshopper optimization algorithm (GOA) was proposed as a hybrid training algorithm for multilayered perceptron neural networks (MLP), and as authors stated in their paper [48], it has obtained promising results. Whale optimization algorithm (WOA) was used in [49] to train ANN for the intrusion detection model, which is able to classify the binary-class, triple-class, and multi-class cyber-attacks and power-system incidents. WOA was also used to train ANN by optimizing the connection weights in [50]. Another recent research, published in [51], utilized a hybrid wolf-bat algorithm for training MLP networks. Swarm intelligence metaheuristics and EA approaches were also used in the domain of CNN hyperparameters' optimization, according to the recent literature survey. The goal of the hyperparameters' optimization is to create an automated framework that will be able to generate either optimal or near-optimal CNN structure for the specific task that needs to be solved. As this task is extremely complex, many researchers tried to optimize only a few CNN hyperparameters, while keeping all other parameters fixed. Two PSO-based approaches for CNNs design were proposed in [52], [53]. The paper [52] presents the improved PSO approach, enhanced with the gradient penalties for generated optimal CNN structures. The authors have validated the proposed approach against three emotional states of subjects which were obtained by using EEG signals and achieved respectable results. On the other hand, in [53], the authors used an orthogonal learning particle swarm optimization (OLPSO) approach to optimize the hyperparameters' values for VGG16 and VGG19 CNNs, and later applied the generated CNNs to diagnose the plant disease. The proposed OLPSO approach was validated against other state-of-the-art algorithms on the same dataset and obtained better classification accuracy.

The problem of the over-fitting was addressed in [54], where authors have implemented and utilized four well-known swarm intelligence algorithms, namely FA, BA, CS, and PSO to establish an adequate selection of the regularization parameter dropout. All four algorithms were validated

on the well-known image classification MNIST dataset and achieved satisfying accuracy.

PSO approach was also used in [55], where authors utilized the canonical PSO for CNN (cPSO-CNN) and managed to adapt to the CNN hyperparameters' variable ranges through the improvement of the canonical PSO exploration capabilities and redefinition of the PSO scalar acceleration coefficients to the vector form. The proposed method was then validated against seven state-of-the-art methods on the same image classification task and proved to be superior both in terms of the classification accuracy and processing costs. Another PSO-based approach was used in [56], where authors managed to create CNNs with a better configuration for a set of five given image classification tasks than AlexNet.

Evolutionary algorithms have also been applied together with CNNs. In [57], the authors proposed an approach which combines CNNs and GA in the case of non-invasive glioma classifications by utilizing the magnetic resonance imaging - MRI. The proposed approach was based on an automatic framework for neuroevolution that utilizes the GA for deep network evolving. Another research, published in [58], focused on generating a DPPN - a differentiable version of the compositional pattern producing network (CPPN). DPPNs was created by utilizing the microbial GA for CNN structure replication. Recently, a new project called DEvol was established by [59]. The goal of the project is to automate deep neural network architecture. DEvol has a support for a variable number of deep and convolutional layers. Available documentation suggests that the proposed framework achieved a test error rate of 0.6% on the MNIST dataset, which is considered to be the state-of-the-art result.

Despite of the fact that numerous algorithms are proposed for the learning process in ANNs, new algorithms should be developed to avoid local minima stagnation, slow convergence, improper exploration-exploitation balance.

III. ANN TRAINING AND HYPERPARAMETER OPTIMIZATION

Neural network training is very important process and plays a crucial role in building a model which will perform better. During the weight learning process, the loss function needs to be optimized. Numerous optimizers have been suggested to address this task, as it can be seen from the recent literature overview. Some of the proposed algorithms for optimization include stochastic gradient descent, Adam, adadelta, adagrad, momentum, and many others [60]–[62].

The common problem with the neural network training process that happens when there is a big difference in the training and test accuracy is called over-fitting. This problem indicates that the network has learned very specific data, and it is not able to correctly predict when the new data is fed to the inputs. To address the problem of the over-fitting, various regularization approaches can be applied, including L_1 and L_2 regularization [63], dropout [64], drop connect [65], batch normalization [66], early stopping, and data augmentation.

Artificial neural networks can be applied to solving difficult problems from various domains. ANNs can obtain respectable results when handling supervised or unsupervised machine learning tasks [67]. These tasks include machine perception problems, where it is not possible to individually interpret the set of available primary features, as stated in [68]. Consequently, ANNs have been intensively utilized for implementation of the pattern recognition, classification, clustering, and predicting problems. For example, in the medical domain, different forms of ANNs were utilized for diagnostics [68] and classification of heart diseases or diabetes. The application of ANNs enabled shortening of the time required of diagnostics by processing large volumes of data during the ANN training.

The most abstract type of ANN is called a single-layer perceptron (SLP). SLPs have only two layers, one input and one output layer, as described in [69]. Unfortunately, this type of ANN is not capable to efficiently process nonlinearly separable patterns, as discussed in [5]. Later, models with multiple layers were proposed, and they are known as multilayer perceptrons (MLP). This kind of neural network overcomes the deficiencies of the SLP model by utilizing one or more hidden layers. MLPs are arguably the most popular form of the ANNs today, with the advantages that include learning capacity, parallel processing, robustness. The most important feature of MLPs is the capacity to generalize [70]. In the research proposed within this paper, MLP with a single hidden layer (SHL) networks are observed, with a goal to optimize the number of hidden units in the hidden layer as well as to optimize the connection weights and biases.

The capabilities of any ANN can be drastically enhanced depending on the chosen learning strategy which was utilized for the network training. Among supervised training techniques, two main approaches exist, namely gradient-based and stochastic methods [5]. The back-propagation is the most widely utilized gradient-descent approach today. It can be applied as an algorithm for the local search, due to the exploitation tendency. However, as the goal is to find the global optimum, the chosen optimizer should be balanced between exploration and exploitation. The exploration phase is necessary to search through the unknown regions of the search space, while the exploitation phase is responsible to focus on the already explored areas. The drawbacks of gradient-based approaches include getting trapped in the local optimum and lazy convergence, to name the few. Therefore, the stochastic optimizers can be utilized for the MLP training, including metaheuristics trainers, which are able to escape from the local optimums.

In the case where it is needed to optimize both the structure of the network and the weights, it is required that the MLP trainer address a large-scale task [5]. As it was discussed in the Section II, both evolutionary and swarm intelligence metaheuristics have been utilized to optimize the connection weights of the network, as well as the MLP's structure and parameters. In the research presented within this paper, we consider both optimizing weights and biases in the SHL

networks, and additionally, to optimize the number of hidden units within the hidden layer.

The MLP networks are a sub-type of feedforward neural networks (FFNN). The FFNNs are formed from a collection of neurons, which perform the role of the processing elements. These neurons are grouped in a series of fully connected layers. MLP consists of three types of parallel layers, namely input, hidden and output layers. Figure 1 shows the MLP architecture with a single hidden layer. Neurons in MLP are set up in a one-directional regime. The layers are connected by the connections with the assigned weights. Every neuron can execute two basic functions: summation and activation. The summation function, that consists of the products of the input values, assigned weights, and bias, is given by the Eq. (1):

$$S_j = \sum_{i=1}^n \omega_{ij} I_i + \beta_j \quad (1)$$

here, n stands for the number of input values, I_i represents the input value i , ω_{ij} is the connection weight, and finally, β_j denotes the bias term.

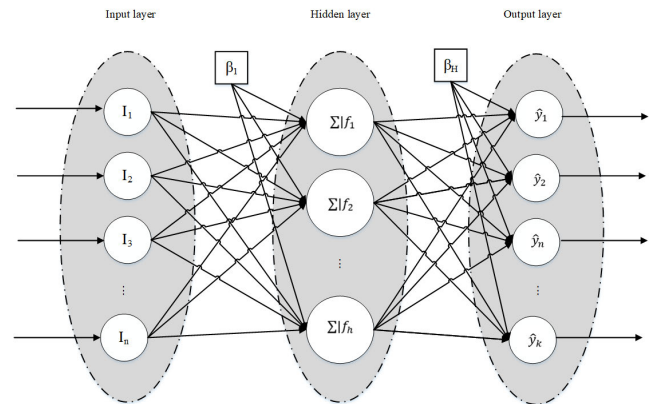


FIGURE 1. MLP network structure with single hidden layer.

The activation function is executed over the output of the Eq. (1). There are several possible types of the activation function, for instance, it is possible to utilize an S-shaped curved sigmoid function, which is given by the Eq. (2):

$$f_j(x) = \frac{1}{1 + e^{-S_j}} \quad (2)$$

The performance of the network is measured by loss function. Common choice of the loss function is the MSE, the mean-squared error loss, which calculates the sum of the squared distances between actual class and the predicted class as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3)$$

For example, if the data has two features, which corresponds to three neurons in the input unit, and if the

hidden layer has three hidden units, the neural network can be represented as:

$$S_1 = \begin{bmatrix} \omega_{11} & \omega_{21} \\ \omega_{12} & \omega_{22} \\ \omega_{13} & \omega_{23} \end{bmatrix} \times \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

IV. PROPOSED METHOD

The ABC algorithm, originally proposed by Karaboga [11], was mainly inspired by the foraging behavior of the swarms of honey bees. ABC was used to solve many challenges, including global [71] and constrained [72] optimization problems, as well as the practical industrial challenges [73]. This section first gives an overview of the basic ABC algorithm, followed by the observed deficiencies and proposed improvements of the basic algorithm. Finally, this section presents the proposed improved ABC approach.

A. BASIC ABC

The original ABC algorithm considers three types of bees: employed, onlookers, and scouts. These bees guide the processes of exploration and exploitation. The artificial bee colony is separated into two parts. One-half of the bees are employed, while the other half are onlookers. Employed bees exploit the sources of food represented by the candidate solutions. On the other side, onlooker bees determine which sources of food to exploit according to the feedback received from the employed bees. If a certain source of food can not be improved in a previously determined number of iterations, the employed bee which was exploiting that source becomes a scout and starts with the exploration process. In the beginning, the ABC algorithm creates an initial population consisted of the solutions distributed in a random fashion [72], by applying the Eq. (4).

$$x_{i,j} = lb_j + rand(0, 1) * (ub_j - lb_j), \quad (4)$$

where $x_{i,j}$ denotes the j -th parameter of the i -th solution, while ub_j and lb_j define the upper and lower borders of the j -th parameter respectively.

In each round of the execution of the algorithm, every employed bee from the population discovers a source of food within its neighborhood, as given with the Eq. (5):

$$v_{i,j} = \begin{cases} x_{i,j} + \phi * (x_{i,j} - x_{k,j}), & R_j < MR \\ x_{i,j}, & otherwise \end{cases} \quad (5)$$

here, $x_{i,j}$ denotes the j -th parameter of the old solution i , $x_{k,j}$ denotes j -th parameter of a neighbor solution k , ϕ denotes a random value in the interval $(0, 1)$, and MR represents the modification rate, a control parameter that prevents convergence to the suboptimal regions of the search space.

After finding a neighborhood solution, its fitness value is compared to the old one, and in case it is better, this new solution is kept in the population.

When the intensification is finished, employed bees will give feedback to the onlooker bees about the quality of the food source. Onlooker bees choose a food source i with a

probability proportional to the fitness value, which can be mathematically modeled by Eq. (6):

$$p_i = \frac{fit_i}{\sum_{j=1}^m fit_j}, \quad (6)$$

where p_i denotes the probability that the food source i will be chosen, m stands for the total amount of food sources, while fit represents the fitness value. The Eq. (6) states that the greater number of onlooker bees will be attracted by the good food sources. After onlooker bees determine which food source will be exploited, they start searching around its neighborhood in the same manner as employed bees, which is described with Eq. (5). If an employed bee is not able to improve certain sources, it will be abandoned, while the bee will become a scout bee. The deserted food source will be replaced by the new, random one. The control parameter *limit* is used to determine which food source (solution) will be deserted. The original ABC can be simplified with the pseudo-code given as Algorithm 1.

Algorithm 1 Original ABC Algorithm

```

Initialization Phase
repeat
    Employed Bees Phase
    Onlooker Bees Phase
    Scout Bees Phase
    Memorize the best solution obtained so far
until iteration = maximum iteration number

```

B. OBSERVED DEFICIENCIES AND PROPOSED IMPROVEMENTS

According to the extensive empirical simulations on standard bound-constrained and unconstrained benchmarks conducted for the purpose of this research, as well as from the results of previous studies [13], [74], the original ABC algorithm suffers from few drawbacks. The algorithm is good in exploration due to the scout bee mechanism, however, the exploitation procedure is not sufficient, therefore basic algorithm does not exhibits good convergence speed. Also, the best solution is not used to guide the algorithm to explore the search space around the current best solution. To overcome the deficiencies of the original ABC, additional two mechanisms are introduced into the basic version.

First, to enhance the exploitation and utilize the information of the current best solution, after 20% of iterations, in subsequent steps, 25% of the worst solutions are replaced by new random solutions created within the boundaries of the minimum and maximum values of the best solutions' components according to Eq. (7). Proposed modification is named guided best bounded (gBestB) mechanism. The values of 20% and 25% were determined empirically by conducting extensive simulations on the benchmark functions, as well as on practical optimization challenges. This behavior could be further adjusted if additional control parameters are employed, however novel parameters would just make harder

for the user to adjust algorithm behavior, therefore they are not included.

$$x_{best} = \min(x_{best}) + r \times (\max(x_{best}) - \min(x_{best})), \quad (7)$$

where the best solution is denoted by x_{best} , the minimum value of the best solution's element is denoted by $\min(x_{best})$, $\max(x_{best})$ indicates to the value of the best solution's element, and r is a random number from the uniform distribution.

Second, to further improve the exploitation capability, the quasi-reflection-based learning (QRL) mechanism [75], which also improves the exploration, and besides that, significantly improves the convergence speed, is incorporated.

Opposite numbers are generated from the solutions as follows:

$$x_j^o = lb_j + ub_j - x_j, \quad (8)$$

where denotes the opposite number of the solution x_j . Parameters lb_j and ub_j denotes the lower bound and upper bound of solution x_j elements.

The quasi-opposite number is calculated as:

$$x_j^{qo} = \text{rnd}\left(\frac{lb_j + ub_j}{2}, x_j^o\right), \quad (9)$$

where the mean of the lower bound and upper bound is calculated by $\frac{lb_j + ub_j}{2}$, $\text{rnd}\left(\frac{lb_j + ub_j}{2}, x_j^o\right)$ generates random number from uniform distribution in range $\left[\frac{lb_j + ub_j}{2}, x_j^o\right]$.

The quasi-reflected component x_j^{qr} is defined as reflection of x_j^{qo} and calculated as:

$$x_j^{qr} = \text{rnd}\left(\frac{lb_j + ub_j}{2}, x_j\right), \quad (10)$$

where the mean of the lower bound and upper bound is calculated by $\frac{lb_j + ub_j}{2}$, $\text{rnd}\left(\frac{lb_j + ub_j}{2}, x_j\right)$ generates random number from uniform distribution in range $\left[\frac{lb_j + ub_j}{2}, x_j\right]$.

In this way, the quasi-reflexive-opposite individuals will be generated, and in case that the original solution is located far away from the optimal value, a fair chance exists that the opposite solution could be located within the area where the optimum is residing.

The QRL mechanism is employed in each iteration in the following way: first, for each solution, its quasi-reflexive-opposite is generated according to Eq. (10) and in this way quasi-reflexive population P^{qr} is created. Afterwards, all solutions from $P \cup P^{qr}$ are sorted based on the fitness in descending order and best NP individuals are propagated to the subsequent iteration. Notations P and NP represent original population and number of individuals in population.

The proposed approach is named as ABCQRBEST, and its pseudo-code, in terms of maximum number of iterations T as termination condition, is presented in Algorithm 2.

The flowchart of the proposed method is presented in Fig. 2.

Algorithm 2 Pseudo-Code of the Proposed ABCQRBEST Algorithm

```

Randomly create the set of the initial population  $P$  of
random  $NP$  solutions
Evaluate the value of the fitness of each solution
while  $t < T$  do
    for  $i = 1$  to  $NP$  do
        Employed bee phase
        Onlooker bee phase
        Scout bee phase
        if  $t \geq T \times 0.2$  then
            Generate random solutions by utilizing Eq. (7)
        end if
    end for
    for  $i = 1$  to  $NP$  do
        for  $j = 1$  to  $D$  do
            if  $x_{i,j} < \frac{lb_j + ub_j}{2}$  then
                 $x_{i,j}^{qr} = x_{i,j} + \left(\frac{lb_j + ub_j}{2} - x_{i,j}\right) \cdot \text{rand}$ 
            else
                 $x_{i,j}^{qr} = \frac{lb_j + ub_j}{2} + \left(x_{i,j} - \frac{lb_j + ub_j}{2}\right) \cdot \text{rand}$ 
            end if
        end for
        Add solution  $x_i^{qr}$  to  $P^{qr}$ 
    end for
    Merge  $P$  and  $P^{qr}$  ( $P \cup P^{qr}$ )
    Evaluate fitness of each solution and sort population
    Choose best  $NP$  solutions for next iteration
end while
Return the best solution
Post-process and visualize results

```

C. COMPLEXITY AND LIMITATIONS OF PROPOSED APPROACH

The most expensive operation in metaheuristics execution is fitness function valuation (FFE). Therefore, based on the most relevant and recent computer science literature, the algorithms' complexity is measured in terms of employed FFEs [15].

Basic ABC algorithm evaluates fitness function in the initialization phase and in the solutions' update phase (employee and onlooker mechanisms). However, due to implementation of QRL mechanism, proposed ABCQRBEST metaheuristics performs additional NP evaluations in each iteration and due to the gBestB mechanism it also utilizes additional $NP \cdot 0.25$ evaluations in $T \cdot 0.8$ iterations. In practice, the scout bee phase is rarely executed and it can be omitted from the complexity calculation.

Therefore, by taking all into account, the complexity of proposed ABCQRBEST metaheuristics in terms of FFEs is given as: $O(NP) + O(2 \cdot NP \cdot T) + O(NP \cdot 0.25 \cdot T \cdot 0.8)$.

Compared with the original ABC, when taking T as termination condition, proposed ABCQRBEST employs higher

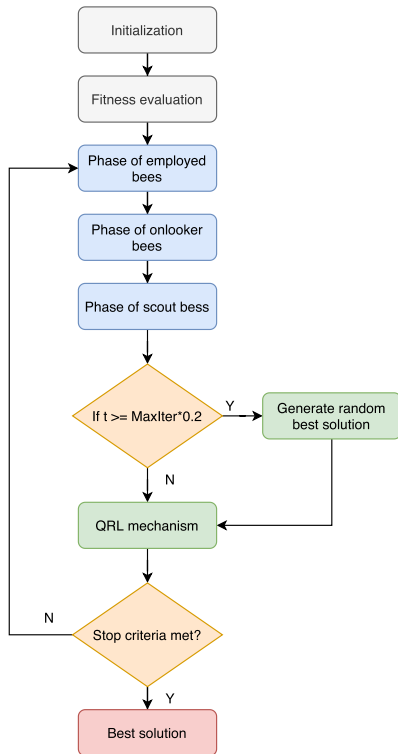


FIGURE 2. ABCQRBEST flowchart.

number of *FFE*s in each iteration, which is one disadvantage of proposed method compared to the original one. However, in practice the number of *FFE*s is taken as termination condition and this drawback does not have influence on the fair comparison between methods.

Also, the another limitation of proposed method stems from the fact that universal control parameters adjustments, that can obtain best performance metrics for all problems (according to the no free lunch theorem - NFL), do not exist. From this context, the percentage of worst solutions that should be replaced with gBestB mechanism and execution point when gBestB is triggered, need to be determined empirically. In this study, optimal values for this behavior are determined and they certainly apply for a vast number of optimization problems.

V. SIMULATIONS AND DISCUSSIONS

A. CEC 2017 BENCHMARK FUNCTIONS SIMULATIONS

The bound-constrained validating process of the suggested ABCQRBEST algorithm was executed on a novel and highly challenging CEC 2017 benchmark function set [76]. The CEC 2017 function set consists of 30 benchmark functions classified into 4 distinctive groups: *F1–F3* represent uni-modal, *F4–F10* represent multi-modal, *F11–F20* represent hybrid functions, and finally benchmarks *F21–F30* represent very complex composite functions. The fourth group of functions has the properties of all uni-modal, multi-modal, and hybrid functions at the same time; additionally, these functions are shifted and rotated.

TABLE 1. CEC 2017 function details.

ID	Name of the function	Class	Search Range	Optimum
F1	Shifted and Rotated Bent Cigar Function	Unimodal	[−100, 100]	100
F2	Shifted and Rotated Sum of Different Power Function	Unimodal	[−100, 100]	200
F3	Shifted and Rotated Zakharov Function	Unimodal	[−100, 100]	300
F4	Shifted and Rotated Rosenbrock's Function	Multimodal	[−100, 100]	400
F5	Shifted and Rotated Rastrigin's Function	Multimodal	[−100, 100]	500
F6	Shifted and Rotated Expanded Scaffer's Function	Multimodal	[−100, 100]	600
F7	Shifted and Rotated Lunacek Bi-Rastrigin Function	Multimodal	[−100, 100]	700
F8	Shifted and Rotated Non-Continuous Rastrigin's Function	Multimodal	[−100, 100]	800
F9	Shifted and Rotated Lévy Function	Multimodal	[−100, 100]	900
F10	Shifted and Rotated Schwefel's Function	Multimodal	[−100, 100]	1000
F11	Hybrid Function 1 (N = 3)	Hybrid	[−100, 100]	1100
F12	Hybrid Function 2 (N = 3)	Hybrid	[−100, 100]	1200
F13	Hybrid Function 3 (N = 3)	Hybrid	[−100, 100]	1300
F14	Hybrid Function 4 (N = 4)	Hybrid	[−100, 100]	1400
F15	Hybrid Function 5 (N = 4)	Hybrid	[−100, 100]	1500
F16	Hybrid Function 6 (N = 4)	Hybrid	[−100, 100]	1600
F17	Hybrid Function 6 (N = 5)	Hybrid	[−100, 100]	1700
F18	Hybrid Function 6 (N = 5)	Hybrid	[−100, 100]	1800
F19	Hybrid Function 6 (N = 5)	Hybrid	[−100, 100]	1900
F20	Hybrid Function 6 (N = 6)	Hybrid	[−100, 100]	2000
F21	Composition Function 1 (N = 3)	Composition	[−100, 100]	2100
F22	Composition Function 2 (N = 3)	Composition	[−100, 100]	2200
F23	Composition Function 3 (N = 4)	Composition	[−100, 100]	2300
F24	Composition Function 4 (N = 4)	Composition	[−100, 100]	2400
F25	Composition Function 5 (N = 5)	Composition	[−100, 100]	2500
F26	Composition Function 6 (N = 5)	Composition	[−100, 100]	2600
F27	Composition Function 7 (N = 6)	Composition	[−100, 100]	2700
F28	Composition Function 8 (N = 6)	Composition	[−100, 100]	2800
F29	Composition Function 9 (N = 3)	Composition	[−100, 100]	2900
F30	Composition Function 10 (N = 3)	Composition	[−100, 100]	3000

Test function *F2* has been removed from the test set as the result of unstable behavior [77]. Fundamental details of CEC 2017 benchmark functions are provided in Table 1.

The simulations were performed with 30-dimensional function variants ($D = 30$), while the results for mean (average) and standard deviation (std) over 50 independent runs were disclosed. The suggested ABCQRBEST algorithm has been validated against famous metaheuristics approaches including the basic FA with dynamic α , cutting-edge enhanced Harris hawks optimization (IHHO) presented in [78], basic Harris hawks optimization (HHO) [79], differential evolution (DE) [80], grasshopper optimization algorithm (GOA) [81], gray wolf optimizer (GWO) [82], moth-flame optimization (MFO) [83], multi-verse optimizer (MVO) [84], particle swarm optimization (PSO) [85], whale optimization algorithm (WOA) [19], sine cosine algorithm (SCA) [86], and the basic version of the ABC ([72]). All algorithms included in comparative analysis were implemented in this study and tested with the parameters suggested in the original publications.

This paper utilizes the same simulation configuration as presented in [78]. The referenced paper [78] published results obtained by utilizing $NP = 30$ and $T = 500$. Since the ABCQRBEST method uses more *FFE* in every run, the maximum number of *FFE* ($maxFFE$) has been set as the termination condition in this study. As every other method in this comparison employs only one *FFE* for each individual in both initialization and update phases, and with a goal to allow valid grounds for fair comparisons, $maxFFE$ was set to $15,030 (NP + NP \cdot T)$.

The proposed ABCQRBEST utilizes the same value for *MR* parameter of 0.8, as suggested for original ABC [11], however, since $maxFFE$ has been taken into the account, value for the *limit* parameter was empirically determined and set to $maxFFE / 2 \cdot NP$ (in this case to 250). The configuration of the control parameters for the opposing methods can be found in [78].

TABLE 2. CEC 2017 comparative analysis results.

Algorithm	F1		F2		F3		F4		F5	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
IHHO	1.86E+2	26.921	n/a	n/a	3.02E+2	52.152	4.03E+2	2.607	5.05E+2	3.251
HHO	1.75E+6	4.29E+5	n/a	n/a	6.71E+2	3.24E+2	4.37E+2	53.631	5.35E+2	24.927
DE	7.54E+7	1.71E+7	n/a	n/a	4.59E+3	1.35E+3	4.29E+2	8.530	5.52E+2	6.232
GOA	1.56E+5	5.24E+4	n/a	n/a	3.05E+2	61.300	4.15E+2	19.48	5.25E+2	16.803
GWO	1.53E+7	4.85E+6	n/a	n/a	3.57E+3	2.77E+3	4.09E+2	10.705	5.19E+2	8.543
MFO	7.17E+6	2.18E+7	n/a	n/a	9.04E+3	9.31E+3	4.20E+2	27.727	5.31E+2	12.860
MVO	1.79E+4	7.99E+3	n/a	n/a	3.05E+2	46.451	4.06E+2	1.392	5.17E+2	9.888
PSO	9.49E+4	8.42E+2	n/a	n/a	3.49E+2	65.409	4.07E+2	10.318	5.26E+2	7.305
WOA	4.27E+7	3.81E+6	n/a	n/a	5.16E+3	4.22E+2	4.61E+2	69.033	5.51E+2	17.46
SCA	1.15E+8	5.91E+7	n/a	n/a	4.03E+3	8.42E+2	4.85E+2	47.271	5.59E+2	9.352
ABC	1.62E+5	3.79E+4	n/a	n/a	3.11E+2	56.814	4.23E+2	21.697	5.31E+2	21.813
ABCQRBEST	1.33E+2	13.578	n/a	n/a	3.02E+2	31.257	4.05E+02	2.488	5.02E+2	3.381
Algorithm	F6		F7		F8		F9		F10	
	mean	STD	mean	STD	mean	STD	mean	STD	mean	STD
IHHO	6.00E+2	0.082	7.49E+2	10.041	8.11E+2	6.526	1.13E+3	85.42	1.69E+3	1.31E+2
HHO	6.38E+2	12.320	7.96E+2	18.921	8.29E+2	5.700	1.44E+3	1.24E+2	2.03E+3	3.42E+2
DE	6.28E+2	4.744	8.01E+2	10.373	8.62E+2	6.873	1.76E+3	1.48E+2	2.09E+3	2.01E+2
GOA	6.08E+2	10.295	7.32E+2	11.375	8.31E+2	14.512	9.97E+2	93.212	1.96E+3	3.17E+2
GWO	6.01E+2	1.909	7.35E+2	16.343	8.16E+2	5.053	9.14E+2	12.11	1.76E+3	3.10E+2
MFO	6.02E+2	2.411	7.46E+2	22.655	8.29E+2	13.786	1.23E+3	2.76E+2	2.02E+3	3.27E+2
MVO	6.03E+2	4.365	7.30E+2	11.278	8.25E+2	12.216	9.00E+2	0.012	1.82E+3	3.60E+2
PSO	6.10E+2	5.539	7.26E+2	9.008	8.19E+2	5.982	9.00E+2	0.003	1.50E+3	2.84E+2
WOA	6.36E+2	13.695	7.82E+2	23.692	8.45E+2	17.470	1.54E+3	3.94E+2	2.19E+3	3.16E+2
SCA	6.24E+2	4.105	7.84E+2	13.299	8.47E+2	7.577	1.03E+3	85.98	2.51E+3	2.18E+2
ABC	6.73E+2	12.415	7.38E+2	14.36	8.33E+2	14.113	9.98E+2	82.17	1.92E+3	2.94E+2
ABCQRBEST	6.00E+2	0.053	7.25E+2	12.414	8.07E+2	6.375	9.98E+2	48.36	1.61E+3	1.26E+2
Algorithm	F11		F12		F13		F14		F15	
	mean	STD	mean	STD	mean	STD	mean	STD	mean	STD
IHHO	1.12E+3	13.523	4.25E+5	3.05E+5	4.42E+3	2.18E+3	1.42E+3	1.651	2.15E+3	5.65E+2
HHO	1.16E+3	45.729	2.56E+6	1.13E+6	1.92E+4	1.16E+4	1.83E+3	2.41E+2	8.63E+3	5.55E+2
DE	1.14E+3	36.317	9.15E+4	6.58E+4	1.35E+3	78.355	1.46E+3	11.826	1.51E+3	18.454
GOA	1.17E+3	58.009	2.24E+6	1.15E+6	1.65E+4	1.13E+4	2.93E+3	1.15E+3	6.48E+3	4.32E+3
GWO	1.34E+3	183.524	1.31E+6	1.54E+6	1.26E+4	7.82E+3	3.19E+3	1.82E+3	5.63E+3	3.16E+3
MFO	1.23E+3	107.133	2.23E+6	4.81E+6	1.61E+4	1.39E+4	8.42E+3	5.42E+3	1.25E+4	1.02E+4
MVO	1.14E+3	27.331	1.52E+6	1.41E+6	9.89E+3	2.55E+3	2.15E+3	1.03E+3	4.05E+3	2.45E+3
PSO	1.10E+3	3.727	4.35E+4	1.26E+4	1.01E+4	7.23E+3	1.49E+3	88.291	1.81E+3	3.75E+2
WOA	1.22E+3	82.415	4.85E+6	2.05E+6	1.57E+4	1.38E+4	3.42E+3	9.82E+2	1.42E+4	9.88E+3
SCA	1.24E+3	96.535	2.41E+7	2.05E+7	6.43E+4	4.69E+4	1.99E+3	4.31E+2	3.21E+3	1.41E+3
ABC	1.18E+3	40.382	2.34E+6	1.23E+6	1.24E+4	1.12E+4	1.90E+3	3.24E+2	3.71E+3	2.15E+3
ABCQRBEST	1.10E+3	1.499	3.20E+4	2.32E+4	1.35E+3	22.539	1.46E+3	22.179	1.51E+3	12.368
Algorithm	F16		F17		F18		F19		F20	
	mean	STD	mean	STD	mean	STD	mean	STD	mean	STD
IHHO	1.73E+3	59.44	1.73E+3	7.519	4.79E+3	1.68E+3	1.90E+3	6.993	2.02E+3	19.561
HHO	1.89E+3	1.47E+2	1.79E+3	65.751	2.02E+4	1.41E+4	1.71E+4	1.21E+4	2.23E+3	86.017
DE	1.69E+3	41.15	1.77E+3	19.514	1.84E+3	23.298	2.75E+3	8.35E+2	2.05E+3	23.711
GOA	1.78E+3	1.76E+2	1.83E+3	1.21E+2	1.63E+4	1.31E+4	3.25E+3	1.95E+3	2.15E+3	74.824
GWO	1.79E+3	1.11E+2	1.77E+3	38.759	2.55E+4	1.84E+4	2.75E+4	2.38E+4	2.09E+3	73.994
MFO	1.85E+3	15.23E+2	1.78E+3	65.311	2.21E+4	1.39E+4	7.81E+3	6.15E+3	2.13E+3	72.321
MVO	1.80E+3	1.44E+2	1.80E+3	46.126	2.03E+4	1.25E+4	4.63E+3	2.62E+3	2.12E+3	86.303
PSO	1.65E+3	65.364	1.72E+3	16.123	7.63E+3	4.46E+3	3.13E+3	2.05E+3	2.06E+3	35.410
WOA	1.96E+3	14.92E+2	1.82E+3	73.459	2.13E+4	1.95E+2	2.07E+5	1.16E+5	2.19E+3	1.11E+2
SCA	1.73E+3	95.425	1.80E+3	25.303E	8.77E+4	9.23E+2	1.15E+4	1.44E+3	2.14E+3	46.855
ABC	1.80E+3	1.75E+2	1.84E+3	1.16E+2	1.71E+4	1.48E+4	3.20E+3	1.58E+3	2.14E+3	72.587
ABCQRBEST	1.72E+3	87.346	1.70E+3	8.698	1.88E+3	22.348	1.90E+3	9.324	2.00E+3	9.978
Algorithm	F21		F22		F23		F24		F25	
	mean	STD	mean	STD	mean	STD	mean	STD	mean	STD
IHHO	2.20E+3	4.615	2.28E+3	17.820	2.59E+3	14.213	2.68E+3	1.31E+2	2.87E+3	85.338
HHO	2.35E+3	53.711	2.32E+3	25.234	2.69E+3	35.522	2.82E+3	93.623	2.95E+3	49.573
DE	2.25E+3	78.104	2.29E+3	17.513	2.63E+3	15.163	2.66E+3	69.502	2.91E+3	15.543
GOA	2.30E+3	56.877	2.38E+3	1.08E+2	2.64E+3	23.536	2.73E+3	57.833	2.93E+3	32.598
GWO	2.30E+3	32.884	2.31E+3	57.573	2.62E+3	13.862	2.74E+3	25.132	2.94E+3	28.256
MFO	2.32E+3	29.255	2.35E+3	93.557	2.63E+3	11.327	2.75E+3	76.435	2.96E+3	37.776
MVO	2.32E+3	11.839	2.33E+3	1.11E+2	2.65E+3	10.445	2.74E+3	18.246	2.92E+3	84.256
PSO	2.27E+3	49.783	2.33E+3	1.03E+2	2.60E+3	72.300	2.70E+3	76.143	2.90E+3	33.735
WOA	2.34E+3	60.021	2.48E+3	2.45E+2	2.66E+3	29.838	2.77E+3	85.902	2.98E+3	1.03E+2
SCA	2.29E+3	65.229	2.41E+3	66.636	2.67E+3	45.449	2.78E+3	11.548	2.98E+3	37.291
ABC	2.31E+3	35.962	2.38E+3	1.12E+2	2.64E+3	18.533	2.74E+3	1.08E+2	2.95E+3	48.058
ABCQRBEST	2.20E+3	49.573	2.25E+3	14.114	2.56E+3	22.627	2.69E+3	1.74E+2	2.83E+3	98.165
Algorithm	F26		F27		F28		F29		F30	
	mean	STD	mean	STD	mean	STD	mean	STD	mean	STD
IHHO	2.93E+3	1.66E+2	3.19E+3	33.657	3.30E+3	48.694	3.20E+3	28.982	2.30E+4	1.45E+4
HHO	3.62E+3	5.39E+2	3.18E+3	51.306	3.41E+3	1.02E+2	3.39E+3	85.653	1.43E+6	1.31E+6
DE	2.95E+3	95.929	3.07E+3	2.558	3.28E+3	27.035	3.21E+3	35.216	3.65E+5	2.31E+5
GOA	3.01E+3	3.65E+2	3.11E+3	25.326	3.31E+3	1.53E+2	3.27E+3	75.411	5.29E+5	3.89E+5
GWO	3.36E+3	5.05E+2	3.10E+3	13.541	3.42E+3	1.33E+2	3.22E+3	49.822	6.17E+5	4.88E+5
MFO	3.05E+3	1.13E+2	3.09E+3	5.722	3.21E+3	93.459	3.26E+3	55.593	6.36E+5	5.93E+5
MVO	3.15E+3	2.77E+2	3.10E+3	21.875	3.36E+3	1.23E+2	3.26E+3	75.139	4.62E+5	4.07E+5
PSO	2.95E+3	2.55E+2	3.12E+3	31.830	3.32E+3	1.35E+2	3.21E+3	62.374	1.13E+6	1.09E+6
WOA	3.37E+3	2.92E+2	3.17E+3	48.124	3.46E+3	1.65E+2	3.46E+3	1.21E+2	1.29E+6	7.53E+5
SCA	3.15E+3	1.82E+2	3.13E+3	13.152	3.38E+3	89.259	3.25E+3	48.339	1.49E+6	9.77E+5
ABC	3.04E+3	2.05E+2	3.12E+3	28.314	3.34E+3	1.18E+2	3.27E+3	33.126	4.75E+5	4.27E+5
ABCQRBEST	2.85E+3	2.47E+2	3.11E+3	50.072	3.15E+3	2.54E+2	3.20E+3	27.959	2.26E+4	1.43E+4

The overall results obtained over the CEC 2017 benchmark functions set are shown in Table 2, where the best results for mean and std indicators for each function are bolded. The results presented in the Table 2 indicate that the ABCQRBEST metaheuristics achieved the best results on 21 benchmark function instances, namely F1, F3, F5, F6, F7, F8, F11, F12, F13, F15, F17, F19, F20, F21, F22, F23, F25, F26, F28, F29, and F30. For some instances, ABCQRBEST achieved the best result, but was tied with results of another method. In such situations, both results

were marked in bold. Generally speaking, the proposed ABCQRBEST method outperformed all other metaheuristics included in the experiments, including the IHHO.

In order to better visualize the stability of algorithms over 50 independent runs, box and whiskers (box plot) diagrams have also been generated, as shown in Fig. 3. The box plot diagrams were generated for randomly chosen functions from the CEC2017 set. The proposed ABCQRBEST was tied for the first place with the IHHO for functions F3, F6, F19, F21, and F29. Also, for some benchmarks, several

functions were tied at the first place, and all were marked in bold. For benchmark F9, the best results for mean metrics were obtained by MVO and PSO algorithms. For benchmark function F9, both basic ABC and proposed ABCQRBEST obtained good scores, and differences between these two algorithms are minimal, except the proposed ABCQRBEST is more stable, as it can be seen that it has lesser std. For this function, WOA and DE obtained poor performances, while PSO obtained the best results, considering the best, worst and mean values that were almost the same. For benchmark F11, ABCQRBEST was tied on the first place with PSO. At the end, for benchmarks F13 and F15, ABCQRBEST shared the first place with DE. Additionally, for F15, ABCQRBEST has better stability than DE.

It can also be noted that ABCQRBEST was outperformed by IHHO algorithm in case of benchmarks F4 and F14. PSO metaheuristics obtained the best results on benchmarks F10 and F16, while the DE method outperformed all other approaches on benchmarks F18, F24 and F27. Considering the F18, the proposed ABCQRBEST has shown better stability than DE, while the WOA has shown the best stability. For benchmark F10, although the PSO obtained the best result, the proposed ABCQRBEST shown the best stability, as the std value was the smallest. Taking all into the account, the suggested ABCQRBEST approach was clearly superior to all competitor methods included in the experiments, justifying the implemented modifications. For benchmark F23, the ABCQRBEST obtained the best results, however not the best stability (dispersion). For F29, the ABCQRBEST obtained the best results for both mean and std, as it can be also seen from the provided box plot diagrams, presented in Fig. 3.

The proposed ABCQRBEST algorithm obtained better stability than the basic ABC, as it can be clearly seen from the presented box plots, where the basic ABC attains greater dispersion between the best and worst runs. However, it is also necessary to note that the basic ABC obtained better results for some benchmark functions than some advanced algorithms such as IHHO.

To demonstrate the statistical significance of the differences between the suggested ABCQRBEST method and all other observed approaches, the statistical Friedman test [87], [88] and the two-way variance analytic by ranks were executed. The results of the observed algorithms on the CEC 2017 function suite for Friedman test rank and the aligned Friedman test rank are shown in Tables 3 and 4.

The findings presented in Table 3 indicate that the suggested ABCQRBEST approach obtained better performances than other popular metaheuristics approaches included in the analysis. The proposed ABCQRBEST achieved the average ranking of 1.551, therefore outperforming the previously best approach IHHO (achieved average ranking of 3.138) by significant margin.

To make sure that presented findings were statistically accurate, the Iman and Davenport test [89] was conducted, as this statistical test has the potential to give improved

TABLE 3. Friedman test rank for the observed methods over 30 CEC 2017 functions.

Function	IHHO	HHO	DE	GOA	GWO	MFO	MVO	PSO	WOA	SCA	FA	ABCQRBEST
F1	2	7	11	5	9	8	3	4	10	12	6	1
F3	1.5	7	10	3.5	8	12	3.5	6	11	9	5	1.5
F4	1	10	9	6	5	8	3	4	11	12	7	2
F5	2	9	11	5	4	8	3	6	10	12	7	1
F6	1.5	11	9	6	3	4	5	7	10	8	12	1.5
F7	8	11	12	4	5.5	3	2	9	10	5.5	1	1
F8	2	6.5	12	8	3	6.5	5	4	10	11	9	1
F9	8	10	12	5.5	3	9	1.5	1.5	11	7	5.5	4
F10	3	9	10	7	4	8	5	1	11	12	6	2
F11	3	6.5	4.5	8	12	10	4.5	1.5	9	11	6.5	1.5
F12	4	10	3	8	5	7	6	2	11	12	9	1
F13	3	11	1.5	10	7	9	4	5	8	12	6	1.5
F14	1	5	3	9	10	12	8	4	11	7	6	2
F15	4	10	1.5	9	8	11	7	3	12	5	6	1.5
F16	5.5	11	3	1	7.5	10	9	2	12	5.5	7.5	4
F17	3	7	4.5	12	4.5	6	8.5	2	10.5	8.5	10.5	1
F18	3	7	1	5	11	10	8	4	9	12	6	2
F19	1.5	10	3	6	11	8	7	4	12	9	5	1.5
F20	2	12	3	10	5	8	6.5	4	11	9	6.5	1
F21	1.5	12	3	7.5	7.5	9.5	9.5	4	11	5.5	5.5	1.5
F22	2	5	3	10	4	8	6.5	6.5	12	11	9	1
F23	2	12	6.5	8	4.5	6.5	9	3	10	11	4.5	1
F24	3	12	1	6	7.5	9	7.5	4	10	11	5	2
F25	2	9	4	4.5	8	10	5	3	11.5	11.5	6.5	1
F26	2	12	3.5	5	10	7	8.5	3.5	11	8.5	6	1
F27	12	11	1	7	5	3	5	8	10	9	5	2
F28	4	10	3	5	11	2	8	6.5	12	9	6.5	1
F29	1.5	11	3.5	10	5	8	3.5	12	6	8	1.5	1
F30	2	11	3	6	7	8	4	9	10	12	5	1
Average Ranking	3.138	9.483	5.362	6.862	6.724	8.017	5.914	4.069	10.621	9.603	6.655	1.551
Rank	2	10	4	8	7	9	5	3	12	11	6	1

TABLE 4. Aligned Friedman test rank results for the analyzed methods over 30 CEC 2017 functions.

Function	IHHO	HHO	DE	GOA	GWO	MFO	MVO	PSO	WOA	SCA	FA	ABCQRBEST
F1	2	7	347	5	9	1	3	4	346	348	6	1
F3	56.5	63	327	58.5	323	334	58.5	61	328	326	60	56.5
F4	144	226	211	177	164	192	152	157	235	278	183	146
F5	138	213	242	194	174	206	169	196	241	245	197	135
F6	153.5	235	218	173	158	161	165	180	232	212	270	153.5
F7	193	264	266	145	155.5	184	140	136	244	246	155.5	131
F8	151	198.5	251	204	167	198.5	191	172	209	231	207	143
F9	141	310	318	89.5	80	285	78.5	78.5	313	91	89.5	87
F10	81	293	301	257	88	290	95	74	309	317	216	75
F11	114	159.5	127.5	185	300	127.5	103.5	103.5	280	159.5	103.5	103.5
F12	13	19	12	17	14	16	15	11	344	345	18	10
F13	43	331	39.5	324	54	321	45	46	314	337	53	39.5
F14	66	70	68	311	316	333	73	69	320	72	71	67
F15	52	329	48.5	322	303	335	68	51	336	55	62	48.5
F16	291.5	308	276	64	298.5	305	302	233	312	291.5	288.5	282
F17	122	225	181.5	269	181.5	205	239.5	113	262.5	239.5	262.5	107
F18	38	76	35	47	332	325	82	41	319	338	50	36
F19	28.5	44	30	33	330	37	34	31	339	42	32	28.5
F20	98	294	112	260.5	150	237.5	223	121	286	247.5	223	96
F21	99.5	281	129	227.5	227.5	252.5	252.5	272	209	209	99.5	101
F22	110	142	118	258	130	217	170.5	297	283	234	101	101
F23	126	279	202.5	223	178.5	202.5	237.5	134	247.5	260.5	178.5	102
F24	119.5	288	105	200.5	220	236	220	132.5	259	267.5	175.5	111
F25	117	243	168	214.5	230	256	195	139	274.5	274.5	214.5	93
F26	84	315	85.5	94	306	106	249.5	85.5	307	249.5	97	77
F27	284	277	119.5	175.5	148	132.5	148	200.5	267.5	220	148	125
F28	137	287	124	166	289	92	254	189.5	296	273	189.5	83
F29	108.5	295	115.5	209	123	187	115.5	304	162.5	187	108.5	108.5
F30	21	342	22	25	26	27	23	340	341	343	24	20
Average Ranking	108.017	221.172	158.621	169.948	188.810	205.259	144.655	122.328	291.724	244.276	147.259	91.931
Rank	2	10	6	7	8	9	4	3	12	11	5	1

statistical conclusions referring to the precision than the χ^2 , as proven and published in [90]. The summarized results of Iman and Davenport's test has been given in Table 5.

After performing the necessary calculations, the final result of the Iman and Davenport test was 36.95, and it has been compared against the F -distribution critical value ($F(9, 9 \times 10) = 1.820$), and it finally shown that the Iman and Davenport test returned a statistically significant higher result. This test also rejected H_0 . Additionally, the Friedman statistics ($\chi_r^2 = 181.50$) are greater than the χ^2 critical value with ten degrees of freedom (1.82), when observing the significance level of $\alpha = 0.05$.

Finally, it allows to reject the null hypothesis (H_0). At the end, it can be concluded that the proposed ABCQRBEST method performed significantly better than other algorithms that were included in tests.

TABLE 5. Friedman and Iman-Davenport statistical test results summary ($\alpha = 0.05$).

Friedman Value	χ^2 Critical Value	p-Value	Iman-Davenport Value	F Critical Value
$1.815 \times 10^{+2}$	$1.968 \times 10^{+1}$	1.110×10^{-16}	$3.695 \times 10^{+1}$	1.820

As the null hypothesis has been rejected by both executed statistical methods, the non-parametric Holm step-down

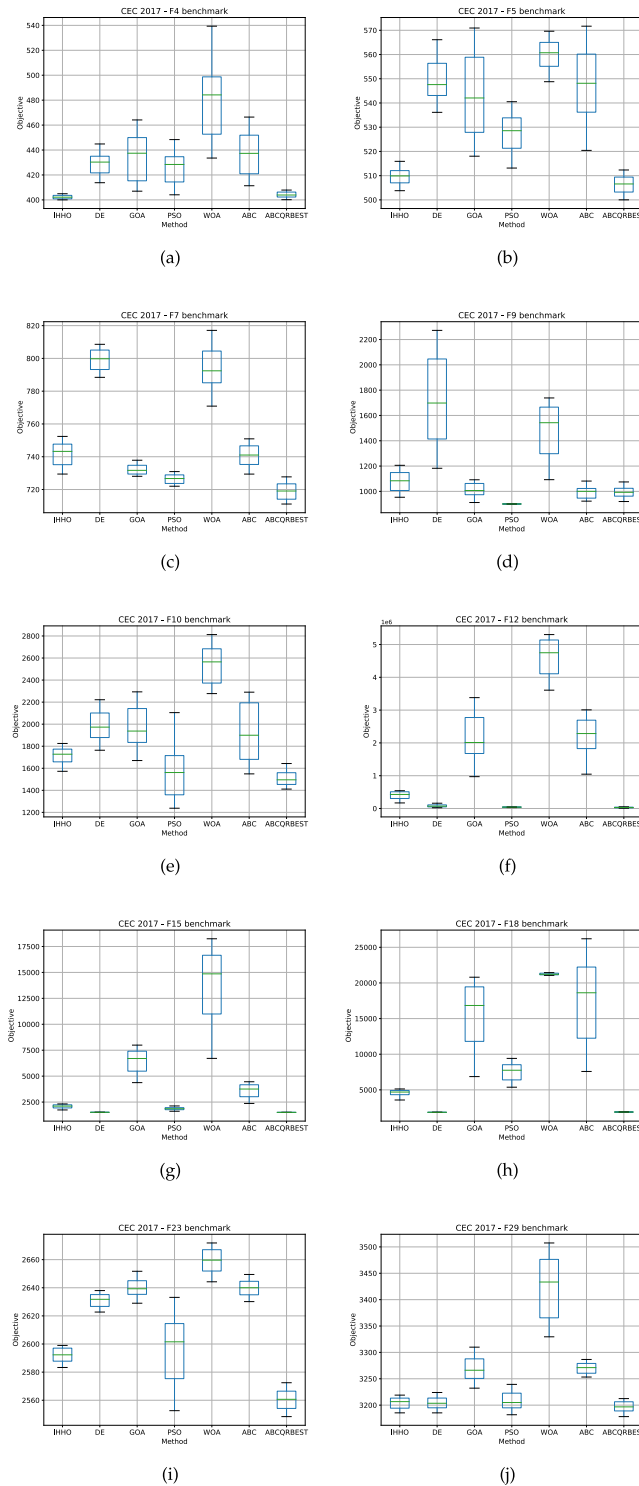


FIGURE 3. Box plot diagrams for ten CEC2017 benchmark functions. a) CEC 2017 – F4 benchmark. b) CEC 2017 – F5 benchmark. c) CEC 2017 – F7 benchmark. d) CEC 2017 – F9 benchmark. e) CEC 2017 – F10 benchmark. f) CEC 2017 – F12 benchmark. g) CEC 2017 – F15 benchmark. h) CEC 2017 – F18 benchmark. i) CEC 2017 – F23 benchmark. j) CEC 2017 – F29 benchmark.

procedure has been executed as well, and the findings are given in Table 6. This approach sorts all algorithms based on their p value and compares with $\alpha/(k - i)$,

where k and i denote the degree of freedom and the algorithm number. For this research, the value of α was set to 0.05 and 0.1.

TABLE 6. Results of the Holm step-down procedure.

Comparison	VALUES	Ranking	alpha = 0.05	alpha = 0.1	H1	H2
ABCQRBEST vs. HHO	0	0	0.00455	0.00909	TRUE	TRUE
ABCQRBEST vs. WOA	0	1	0.00500	0.01000	TRUE	TRUE
ABCQRBEST vs. SCA	0	2	0.00556	0.01111	TRUE	TRUE
ABCQRBEST vs. MFO	4.29×10^{-12}	3	0.00625	0.01250	TRUE	TRUE
ABCQRBEST vs. GOA	1.02×10^{-8}	4	0.00714	0.01429	TRUE	TRUE
ABCQRBEST vs. GWO	2.35×10^{-8}	5	0.00833	0.01667	TRUE	TRUE
ABCQRBEST vs. FA	3.53×10^{-8}	6	0.01000	0.02000	TRUE	TRUE
ABCQRBEST vs. MVO	2.04×10^{-6}	7	0.01250	0.02500	TRUE	TRUE
ABCQRBEST vs. DE	2.86×10^{-5}	8	0.01667	0.03333	TRUE	TRUE
ABCQRBEST vs. PSO	3.92×10^{-3}	9	0.02500	0.05000	TRUE	TRUE
ABCQRBEST vs. IHHO	4.69×10^{-2}	10	0.05000	0.10000	FALSE	FALSE

The findings shown in the Table 6 indicate that the suggested ABCQRBEST method significantly outperformed every other opposed algorithm, except state of the art IHHO, at both significance levels.

B. ANN TRAINING AND HYPERPARAMETER OPTIMIZATION EXPERIMENT

This subsection first describes the datasets used in this experiment, following by the description of metrics used for the proposed ABCQRBEST algorithm evaluations. Next, the adaptations of method for ANN training are elaborated, along with the setup of neural network weight optimization experiment, comparative analysis and obtained results' interpretation. Finally, at the end of this section, the hidden unit and weight optimization experiment's setup and results are described.

Experiment design in terms of employed datasets, pre-processing, metrics utilized for comparison is the same as in the study proposed in [48].

1) DATASETS DESCRIPTION

The proposed optimization algorithm is tested on the following five different medical datasets for binary classification:

- Breast cancer dataset;
- Parkinson dataset;
- Diabetes dataset;
- SAheart dataset and
- Vertebral dataset

All datasets are freely available and downloadable from globally recognized UCL machine learning repository [91] and all of them have two classes, while the number of features and instances varies from one to the other. Details of employed datasets are summarized in Table 7.

TABLE 7. Dataset details.

Dataset	Classes	Features	Instances
Breast cancer dataset	2	9	699
Parkinson dataset	2	22	195
Diabetes dataset	2	8	768
SAheart dataset	2	9	462
Vertebral dataset	2	6	310

The breast cancer dataset [92], [93] was created by Dr. William H. Wolberg from the University of Wisconsin

Hospitals, Madison. The dataset has two classes, one class is indicating to benign and another class to the malignant cancer diagnosis. The total number of instances is 699, and each instance is presented by 9 numerical features (Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, Mitoses).

The Parkinson dataset [94] was created at Oxford University by Max Little. Each column in the table is a particular voice measure, and each row corresponds to one of 195 voice recordings from the individuals. The dataset has two classes, one class is indicating to a healthy patient and the other class to a patient diagnosed with Parkinson's diseases. Each instance in the dataset is presented by 22 numerical features.

The diabetes dataset is created by the National Institute of Diabetes, Digestive, and Kidney Diseases. The dataset has two classes, indicating whether the patient is diagnosed with diabetes or not. The total number of instances is 768, and each instance is presented by 8 numerical features, including the number of pregnancies, glucose, blood pressure, skin thickness, insulin level, BMI, diabetes pedigree function, and age.

The SAheart (South African Heart Disease) dataset [95] contains the data of a heart-disease high-risk region of the Western Cape, South Africa. The samples in the observations are only of males. The dataset has two classes, indicating the person has coronary heart disease (CHD) or not. The total number of instances is 462, and each instance is presented by 9 numerical features, such as systolic blood pressure, cumulative tobacco, low-density lipoprotein cholesterol, adiposity, obesity, current alcohol consumption, family history of heart disease, type-A behavior, and age.

The Vertebral dataset is created by Dr. Henrique da Mota. The dataset has two classes, classifying orthopedic patients to normal and abnormal (patients having Disk Hernia or Spondylolisthesis). The total number of instances is 310, and each sample is presented by 6 biomechanical features.

The feature distributions of five datasets are presented in Fig. 4.

2) CLASSIFICATION METRICS

The performance of proposed approach is evaluated on various standard metrics, such as accuracy, specificity, sensitivity, geometric mean (g-mean) and area under the curve (AUC). The metrics are calculated by using the following expressions:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (11)$$

$$specificity = \frac{TN}{TN + FP} \quad (12)$$

$$sensitivity = \frac{TP}{FP + TP} \quad (13)$$

$$g - mean = \sqrt{specificity \times sensitivity} \quad (14)$$



where TP denotes true positive, TN the true negative, FP the false positive, and FN the false negative values from the confusion matrix. The confusion matrix is presented in Fig 5.

prediction of negative samples out of all negative actual values (TN and FP). Conversely, sensitivity measures correct prediction of positive samples out of all positive actual values. The geometric mean is calculated by taking the square root of the product of specificity and sensitivity. The AUC is a very important measure in machine learning and it provides an aggregated performance measure of all possible classification thresholds, in other words, it tells how much the method is able to make a difference between classes.

		Actual values	
		positive	negative
Predicted values	positive	TP	FP
	negative	FN	TN

FIGURE 5. Confusion Matrix.

3) PROPOSED ABCQRBEST ADAPTATIONS FOR ANN TRAINING

In the neural network training, the ABCQRBEST optimizes the values of the weights in the hidden units. In the algorithm, one solution encodes the weights and biases of the neural network. Thus, the solution vector consists of the connection weights and biases between the input layer and the hidden layer, as well as the connection weights and biases between the hidden layer and output layer.

The procedure of the neural network training by ABCQRBEST can be described as follows:

- Step 1: Initialize the solutions of N neural network randomly.
- Step 2: Define the fitness function (MSE)
- Step 3: Evaluate the fitness value of each solution (neural network)
- Step 4: Update the solution vector of each network by ABCQRBEST
- Step 5: Evaluate the fitness value of each solution (neural network)
- Step 6: Save the current best solution with minimum error rate
- Step 7: After the termination criteria met, return the best solution
- Step 8: Test the best network with the test dataset.

Algorithm 3 describes the pseudocode of the neural network training by the proposed ABCQRBEST approach.

The flowchart of the proposed method on neural network training is presented in Fig. 6.

4) NEURAL NETWORK WEIGHT OPTIMIZATION EXPERIMENT

In the neural network weight optimization experiment, the ABCQRBEST proposed approach is used to optimize the values of weights and biases in the hidden units. Each solution encodes the weights and biases of the neural network. Thus, the solution vector consists of the connection weights and biases between the input layer and the hidden layer, as well as the connection weights and biases between the hidden layer and output layer. Therefore, dimension of a solution is calculated as follows:

$$W = (I \times H + H) + (H \times O + O) \quad (16)$$

Algorithm 3 Pseudocode of the Neural Network Training by ABCQRBEST

```

Define the size of the population  $N$ 
Randomly initialize the connection weights and biases  $x_i (i = 1, 2, \dots, n)$ 
Define the fitness function  $f(x)$ 
Define counter  $t = 0$ 
Define maximum number of iterations  $MaxIter$ 
while  $t < MaxIter$  do
  for each solution do
    Set the connection weights and biases of the neural network with solution's decision variables
    Evaluate the fitness for current solution
    Optimize decision variables by ABCQRBEST
  end for
  Sort the networks in the population by fitness function
  Save the current best solution with minimum error rate
end while
return the best solution and test the best network with the test dataset.

```

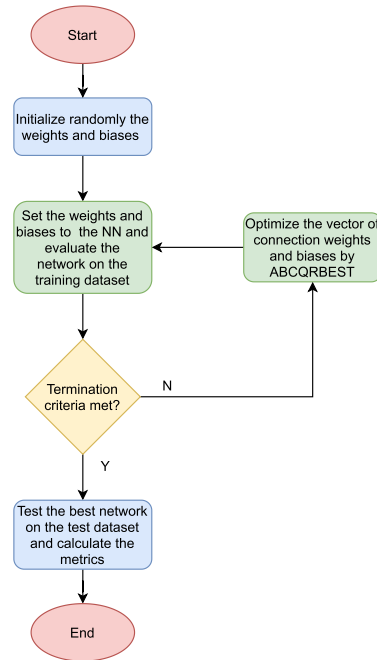


FIGURE 6. ABCQRBEST flowchart for ANNs training.

where W denotes the one-dimensional vector of weights and biases, I represents the number of input features in the input layer, H indicates the number of hidden units in the hidden layer, and O denotes the output layer, which consists of two nodes in all conducted experiments since all datasets fall into the group of binary classification challenges.

For the purpose of conducted study five medical datasets are split into training and testing data. The training data consists of 2/3, while the testing encompass 1/3 of all observations. In data pre-processing phase, with the goal of adjusting the influence of each feature on classification performance,

data normalization on all features in each dataset, by changing the range of the data between 0 and 1, is applied, according to the following formula:

$$X_{norm} = \frac{X_i - X_{min}}{X_{max} - X_{min}}, \quad (17)$$

where X_i is the i -th feature, the minimum and the maximum value of features are X_{min} and X_{max} , respectively. The normalized feature data is denoted by X_{norm} .

In this work, the neural network model has only one hidden layer and the number of hidden units in the layer depends on the number of features of the corresponding dataset. The hidden unit number is calculated as:

$$H = 2 \times I + 1, \quad (18)$$

where the number of hidden units is denoted by H , and I indicates to the number of input features in a given dataset.

The fitness function of the algorithm is given as the mean-squared error (MSE) loss:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2, \quad (19)$$

where the total number of instances are denoted by n , the y represents actual value, while the predicted value is represented by $\hat{y}$.

In the initialization phase, the population of N individuals with W components is generated randomly within the boundaries $[-1, 1]$ for each parameter (weight or bias) of every solution. The experimental setup is similar to the one conducted in [48], where population of 50 individuals ($NP = 50$) is iterated during the course of 250 iterations ($T = 250$). However, to make objective comparative analysis, as in the case of CEC 2017 benchmark simulations, instead of T , the $maxFFE$ was set as the termination condition. The $maxFFE$ is calculated by using expression $NP + NP \cdot T$, which in this case yields the total of 12,550 FFE in one run.

Obtained results of ABCQRBEST metaheuristics are compared to other algorithms, which are evaluated against same datasets in [48], namely the GOA, basic genetic algorithm (GA), PSO, ABC, flower pollination algorithm (FPA) [96], bat algorithm (BAT/BA) [17], firefly algorithm (FF/FA) [15], monarch butterfly optimization (MBO) [97] and biogeography-based optimization (BBO) [98]. All these methods were also implemented and tested in this study and similar results as in [48] were obtained, therefore validity of the study from [48] was confirmed. All obtained results are generated over 30 independent runs

However, on top of the above mentioned approaches employed in comparative analysis, with the goal of conducting wider and more rigid evaluation of proposed ABCQRBEST, other well-known metaheuristics gray wolf optimization (GWO) [82], fruit fly optimization algorithm (FOA) [99], whale optimization algorithm (WOA) [19], salp swarm algorithm (SSA) [100] and brain storm optimization algorithm (BSOA) [101] were also implemented for the same problem and its results are included in comparison tables.

All methods implemented for the purpose of comparative analysis were tested with the same control parameters' setup as suggested in original studies and devised ABCQRBEST method was tested with the same parameters as in CEC2017 experiments (Subsection V-A).

The experimental results on the five datasets are presented in tables 8, 9, 10, 11, 12, while the results for each method are summarized in Table 13. The statistical results in the tables includes the average, standard deviation, specificity, sensitivity, g-mean and AUC. Results highlighted with bold are indicating the best results.

To provide better insights into the methods performance, mean classification error convergence speed graphs for some better performing algorithms are depicted in Figure 7.

TABLE 8. Breast cancer dataset results.

Algorithms	Result	Accuracy	Specificity	Sensitivity	G-Mean	AUC
ABCQRBEST	Average	0.97588	0.98811	0.97810	0.97259	0.99740
	StdDev	0.00580	0.00677	0.00869	0.00785	0.00294
	Best	0.98346	1.00000	0.98500	0.98073	0.99874
	Worst	0.96491	0.97315	0.97250	0.96787	0.98524
ABC	Average	0.96891	0.96214	0.97240	0.96713	0.98544
	StdDev	0.00793	0.02592	0.00890	0.01161	0.01499
	Best	0.98319	1.00000	0.98726	0.98718	0.99772
	Worst	0.94958	0.91358	0.94904	0.94047	0.92593
GOA	Average	0.97115	0.95309	0.98047	0.96665	0.99536
	StdDev	0.00514	0.01352	0.00553	0.00675	0.00105
	Best	0.97899	0.97531	0.98726	0.97810	0.99693
	Worst	0.95798	0.92593	0.96178	0.94991	0.99229
GA	Average	0.96751	0.94362	0.97983	0.96151	0.99549
	StdDev	0.00583	0.01796	0.00294	0.00884	0.00072
	Best	0.97479	0.97531	0.98726	0.97492	0.99709
	Worst	0.95378	0.90123	0.97452	0.94022	0.99418
PSO	Average	0.97045	0.94774	0.98217	0.96471	0.99558
	StdDev	0.00752	0.02331	0.00565	0.01126	0.00100
	Best	0.97899	0.98765	0.99363	0.98107	0.99756
	Worst	0.95378	0.90123	0.96815	0.94022	0.99402
FPA	Average	0.97241	0.95432	0.98174	0.96789	0.99511
	StdDev	0.00621	0.01838	0.00277	0.00922	0.00090
	Best	0.98319	0.98765	0.98726	0.98427	0.99670
	Worst	0.95378	0.90123	0.97452	0.94022	0.99292
BAT	Average	0.96218	0.93457	0.97643	0.95494	0.97079
	StdDev	0.01422	0.04541	0.00805	0.02250	0.02875
	Best	0.98319	0.98765	0.99363	0.98127	0.99568
	Worst	0.92437	0.79012	0.96178	0.88605	0.89188
FF	Average	0.97311	0.98089	0.95802	0.96938	0.99619
	StdDev	0.00324	0.00000	0.00951	0.00481	0.00024
	Best	0.97899	0.98089	0.97531	0.97810	0.99678
	Worst	0.96639	0.98089	0.93827	0.95935	0.99575
MBO	Average	0.96695	0.94074	0.98047	0.96031	0.99453
	StdDev	0.00770	0.02302	0.00708	0.01115	0.00245
	Best	0.97899	0.98765	0.99363	0.98107	0.99693
	Worst	0.94958	0.87654	0.95541	0.93026	0.98616
BBO	Average	0.97255	0.95514	0.98153	0.96822	0.99550
	StdDev	0.00478	0.01316	0.00256	0.00678	0.00082
	Best	0.98319	0.97531	0.98726	0.98127	0.99670
	Worst	0.96639	0.93827	0.97452	0.95935	0.99355
GWO	Average	0.97124	0.95492	0.97895	0.966213	0.99481
	StdDev	0.004799	0.01321	0.00737	0.00746	0.00095
	Best	0.98247	0.96982	0.98346	0.98249	0.99668
	Worst	0.96689	0.93984	0.97495	0.95986	0.99324
FOA	Average	0.96941	0.95113	0.97683	0.965417	0.99245
	StdDev	0.00522	0.01329	0.00298	0.00723	0.00085
	Best	0.97324	1.00000	0.98256	0.98015	0.99558
	Worst	0.96378	0.93982	0.97215	0.95876	0.99240
WOA	Average	0.97113	0.95967	0.97863	0.96745	0.99568
	StdDev	0.00495	0.01457	0.00273	0.00710	0.00098
	Best	0.98299	1.00000	0.98465	0.98224	0.99694
	Worst	0.96895	0.94117	0.97486	0.95987	0.99294
SSA	Average	0.97334	0.95497	0.98003	0.96945	0.99104
	StdDev	0.00514	0.01359	0.00279	0.00731	0.00079
	Best	0.98328	1.00000	0.98598	0.98354	0.99714
	Worst	0.96813	0.94106	0.97517	0.95948	0.99224
BSOA	Average	0.96545	0.94997	0.97983	0.96542	0.99425
	StdDev	0.00526	0.01568	0.00398	0.00754	0.00096
	Best	0.98057	0.96948	0.98154	0.97864	0.99558
	Worst	0.95973	0.92947	0.96943	0.95882	0.99142

The proposed method achieved the best results on average accuracy, specificity, g-mean, and AUC on the breast

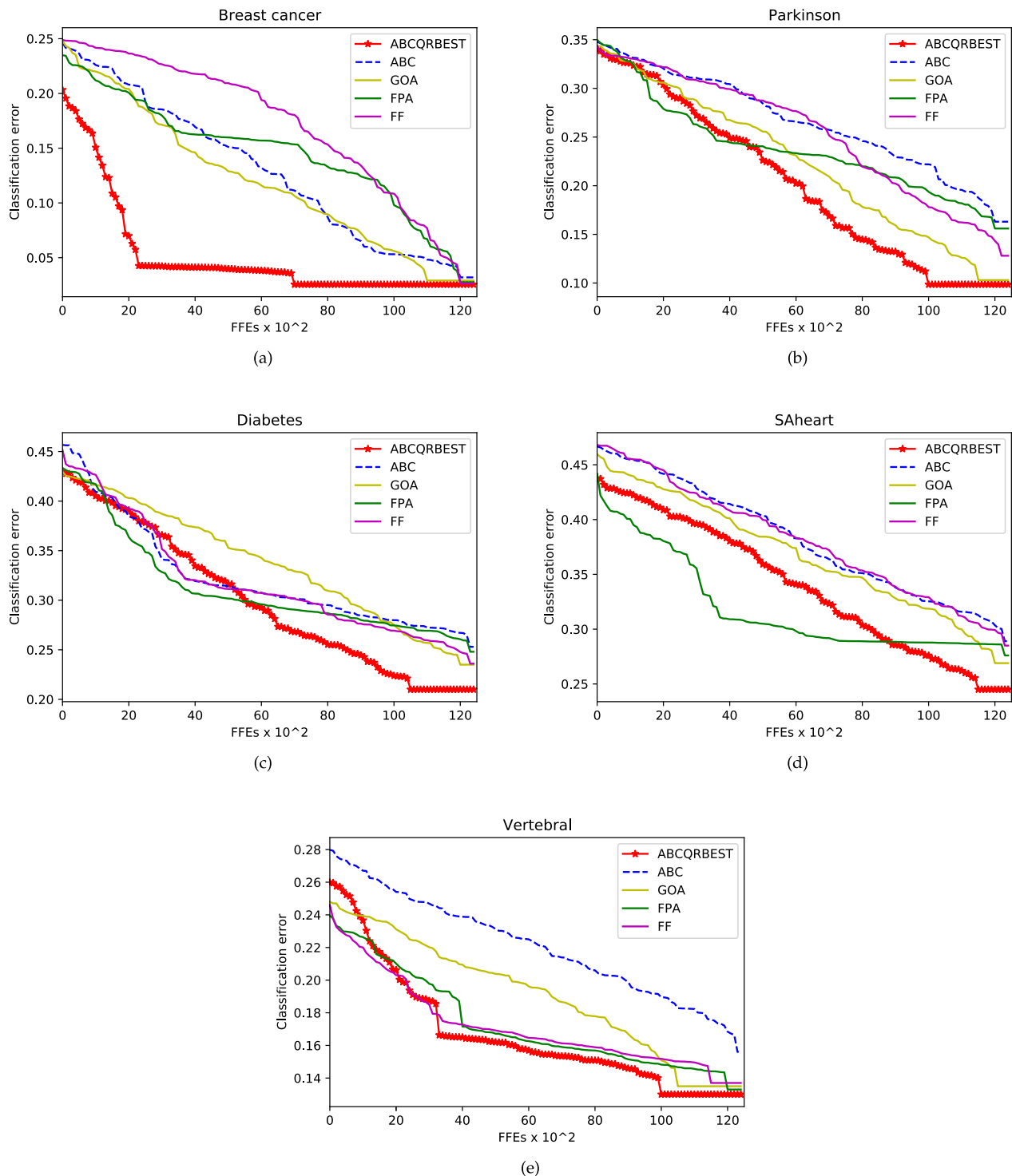


FIGURE 7. Mean classification error convergence speed graphs. a) Breast cancer classification error convergence. b) Parkinson classification error convergence. c) Diabetes classification error convergence. d) SAheart classification error convergence. e) Vertebral classification error convergence.

cancer dataset test. ABCQRBEST shows high performance on the accuracy, specificity, and AUC, and the g-mean metrics. In the case of the standard deviation, the FF resulted in the best values on most metrics. In the test on the Parkinson dataset, ABCQRBEST resulted in best values on

the accuracy, g-mean, and AUC. FF and BBO have the best performance in specificity and sensitivity, respectively. On the diabetes dataset, the ABCQRBEST shows the best performance on the accuracy and g-mean, the second-best performing algorithm is FF and the third is GOA. In the

TABLE 9. Parkinson dataset results.

Algorithms	Result	Accuracy	Specificity	Sensitivity	G-Mean	AUC
ABCQRBEST	Average	0.91176	0.81452	0.97131	0.86859	0.95239
	StdDev	0.03260	0.13177	0.03620	0.07351	0.02230
	Best	0.97769	1.00000	1.00000	0.99491	0.99632
	Worst	0.83923	0.64286	0.86538	0.77592	0.92195
ABC	Average	0.83831	0.43750	0.96405	0.64645	0.80098
	StdDev	0.01842	0.08041	0.02962	0.05034	0.09376
	Best	0.86567	0.68750	1.00000	0.77015	0.92525
	Worst	0.79104	0.31250	0.86275	0.55351	0.56985
GOA	Average	0.90746	0.72500	0.96471	0.83084	0.91891
	StdDev	0.03601	0.15536	0.01885	0.09294	0.08401
	Best	0.97015	1.00000	1.00000	0.98020	0.99387
	Worst	0.82090	0.43750	0.92157	0.64169	0.68873
GA	Average	0.88109	0.60000	0.96928	0.75794	0.92823
	StdDev	0.02055	0.12128	0.03364	0.06631	0.02968
	Best	0.92537	0.87500	1.00000	0.87867	0.96201
	Worst	0.83582	0.43750	0.88235	0.66144	0.84069
PSO	Average	0.84478	0.46042	0.96536	0.66365	0.85118
	StdDev	0.03026	0.08605	0.03203	0.06245	0.03778
	Best	0.89552	0.62500	1.00000	0.75000	0.90319
	Worst	0.76119	0.31250	0.88235	0.53665	0.70833
FPA	Average	0.86070	0.50417	0.97255	0.69764	0.89011
	StdDev	0.02458	0.08675	0.02280	0.05833	0.04068
	Best	0.91045	0.68750	1.00000	0.81274	0.95343
	Worst	0.80597	0.37500	0.92157	0.60634	0.80025
BAT	Average	0.86269	0.55625	0.95882	0.72112	0.84641
	StdDev	0.04156	0.16926	0.03865	0.10801	0.11313
	Best	0.95522	0.93750	1.00000	0.90749	0.98284
	Worst	0.76119	0.25000	0.82353	0.50000	0.62500
FF	Average	0.87214	0.98693	0.50625	0.70564	0.88039
	StdDev	0.02025	0.01296	0.06218	0.04496	0.01810
	Best	0.91045	1.00000	0.62500	0.79057	0.91176
	Worst	0.83582	0.96078	0.43750	0.64834	0.83701
MBO	Average	0.85075	0.47083	0.96993	0.67160	0.85633
	StdDev	0.03208	0.10346	0.02168	0.07894	0.04241
	Best	0.89552	0.62500	1.00000	0.77491	0.91544
	Worst	0.76119	0.25000	0.90196	0.49507	0.73775
BBO	Average	0.88159	0.55208	0.98497	0.73480	0.93656
	StdDev	0.02446	0.09297	0.01517	0.06249	0.02311
	Best	0.92537	0.75000	1.00000	0.85749	0.98039
	Worst	0.82090	0.37500	0.94118	0.61237	0.89951
GWO	Average	0.86734	0.61236	0.92142	0.72546	0.84566
	StdDev	0.03145	0.12454	0.04591	0.07892	0.12044
	Best	0.89425	0.74594	1.00000	0.82246	0.96221
	Worst	0.78689	0.65883	0.91453	0.68521	0.81224
FOA	Average	0.83513	0.62887	0.91553	0.71545	0.85122
	StdDev	0.03588	0.12254	0.04982	0.07936	0.13045
	Best	0.88451	0.73674	0.98500	0.85472	0.95984
	Worst	0.76548	0.65423	0.90259	0.68348	0.82453
WOA	Average	0.84564	0.62345	0.92198	0.71537	0.83242
	StdDev	0.03984	0.12559	0.04786	0.08045	0.10157
	Best	0.88456	0.73544	0.95250	0.81124	0.91216
	Worst	0.77519	0.66572	0.90276	0.67523	0.78624
SSA	Average	0.83546	0.62103	0.91547	0.70254	0.82584
	StdDev	0.04255	0.12148	0.04983	0.08561	0.09554
	Best	0.88552	0.75688	1.00000	0.81257	0.92456
	Worst	0.77543	0.65442	0.90245	0.66573	0.79246
BSOA	Average	0.84256	0.63584	0.90458	0.71567	0.84598
	StdDev	0.04042	0.11204	0.04331	0.07975	0.07856
	Best	0.90512	0.76547	1.00000	0.81253	0.91225
	Worst	0.75878	0.63456	0.90254	0.69874	0.80545

heart test results, ABCQRBEST shows the best average, best, and worst statistical results on the accuracy and AUC. In the experimental results of the fifth dataset, the Vertebral dataset, the proposed method has the highest accuracy, sensitivity, g-mean, and AUC.

Summarizing the obtained experimental results, ABCQRBEST has 46 best values in total, and the second method is FF with a total of 23 best statistical results. In the case of the ABCQRBEST, the best values are achieved on average and best, while in the case of FF, the best results are achieved on the standard deviation.

From mean classification error rate convergence speed graphs shown in Figure 7, some important conclusions regarding the time complexity in terms of *FFE* can be derived. It is observed that the proposed ABCQRBEST

TABLE 10. Diabetes dataset results.

Algorithms	Result	Accuracy	Specificity	Sensitivity	G-Mean	AUC
ABCQRBEST	Average	0.79103	0.84169	0.81278	0.82673	0.80797
	StdDev	0.00859	0.01475	0.03692	0.01375	0.03422
	Best	0.80625	0.86444	0.89222	0.85225	0.87050
	Worst	0.78281	0.81408	0.74105	0.80037	0.76905
ABC	Average	0.74822	0.84799	0.57569	0.69560	0.82251
	StdDev	0.01983	0.04606	0.08174	0.03950	0.01646
	Best	0.80534	0.94578	0.70833	0.78115	0.86509
	Worst	0.70992	0.72892	0.37500	0.59554	0.78181
GOA	Average	0.76489	0.88373	0.55938	0.70303	0.85528
	StdDev	0.00836	0.00674	0.01645	0.01138	0.00602
	Best	0.77863	0.89157	0.59375	0.72511	0.86433
	Worst	0.75191	0.87349	0.54167	0.68785	0.84682
GA	Average	0.75165	0.85723	0.56910	0.69804	0.83888
	StdDev	0.01207	0.01653	0.03378	0.01890	0.00782
	Best	0.78244	0.89759	0.63542	0.73016	0.85950
	Worst	0.72137	0.83133	0.47917	0.65582	0.82674
PSO	Average	0.73104	0.86265	0.50347	0.65648	0.81890
	StdDev	0.02475	0.04603	0.07088	0.03879	0.01773
	Best	0.78244	0.95181	0.62500	0.73887	0.84419
	Worst	0.67176	0.79518	0.37500	0.57216	0.77987
FPA	Average	0.75242	0.86546	0.55694	0.69369	0.83299
	StdDev	0.01486	0.02183	0.03847	0.02173	0.00894
	Best	0.78244	0.92169	0.62500	0.73376	0.84752
	Worst	0.72137	0.82530	0.48958	0.65620	0.81859
BAT	Average	0.76374	0.87289	0.57500	0.70779	0.84349
	StdDev	0.01405	0.01792	0.04329	0.02411	0.01629
	Best	0.79389	0.90361	0.64583	0.75367	0.86383
	Worst	0.73282	0.81325	0.48958	0.66068	0.79744
FF	Average	0.76349	0.57153	0.87450	0.70695	0.84664
	StdDev	0.00465	0.00595	0.00594	0.00471	0.00328
	Best	0.77481	0.58333	0.88554	0.71873	0.85398
	Worst	0.75191	0.56250	0.86145	0.69611	0.83967
MBO	Average	0.74733	0.85763	0.55660	0.68939	0.82938
	StdDev	0.02026	0.03450	0.05903	0.03197	0.01912
	Best	0.78626	0.95783	0.64583	0.73802	0.86195
	Worst	0.70611	0.78916	0.42708	0.60867	0.77548
BBO	Average	0.75611	0.86325	0.57083	0.70183	0.84211
	StdDev	0.01062	0.01638	0.01823	0.01136	0.00744
	Best	0.77099	0.89157	0.60417	0.72143	0.85442
	Worst	0.73282	0.83133	0.53125	0.67885	0.82649
GWO	Average	0.76542	0.87546	0.58634	0.72546	0.83587
	StdDev	0.01145	0.01752	0.01785	0.01136	0.00756
	Best	0.77246	0.89214	0.62456	0.73298	0.85347
	Worst	0.72987	0.82976	0.53228	0.68045	0.82873
FOA	Average	0.77245	0.86875	0.56851	0.70054	0.83975
	StdDev	0.01154	0.01785	0.01792	0.01054	0.00893
	Best	0.77642	0.89364	0.60978	0.72457	0.85986
	Worst	0.73923	0.84512	0.54657	0.68014	0.83042
WOA	Average	0.75313	0.85452	0.56984	0.71125	0.83547
	StdDev	0.01146	0.01685	0.01907	0.01214	0.00805
	Best	0.77114	0.89221	0.60224	0.72059	0.85268
	Worst	0.73117	0.83564	0.54235	0.68042	0.82146
SSA	Average	0.74526	0.85475	0.56589	0.70147	0.83986
	StdDev	0.01245	0.01741	0.01998	0.01004	0.00798
	Best	0.78041	0.89452	0.61471	0.73128	0.85354
	Worst	0.72457	0.82989	0.53986	0.68525	0.82193
BSOA	Average	0.73984	0.84258	0.58452	0.71575	0.83154
	StdDev	0.01268	0.01785	0.01963	0.01854	0.00806
	Best	0.76895	0.88452	0.59458	0.71997	0.84213
	Worst	0.72147	0.82043	0.52984	0.67312	0.81244

algorithm for all datasets obtains best reported accuracy (classification error) after between 50% and 90% of *maxFFE*, while all other approaches converge throughout the whole run. This means that the ABCQRBEST can reach best reported results within smaller number of *FFE*, therefore computation time compared to other state-of-the-art methods is reduced. To further test this claim, all experiments for ANN training are executed again with only 10,000 *FFE* and it is observed that the same results as shown in tables 8 - 12 are obtained.

5) HIDDEN UNITS AND WEIGHT OPTIMIZATION IN NEURAL NETWORKS

The hidden units and weight optimization in neural networks are an extension of the previous neural network weigh

TABLE 11. SAheart dataset results.

Algorithms	Result	Accuracy	Specificity	Sensitivity	G-Mean	AUC
ABCQRBEST	Average	0.75520	0.80108	0.82331	0.70359	0.79639
	StdDev	0.01593	0.01076	0.01925	0.01716	0.01076
	Best	0.79254	0.84059	0.87730	0.74657	0.82560
	Worst	0.73409	0.79570	0.80730	0.68118	0.77987
ABC	Average	0.71160	0.82276	0.49753	0.63644	0.74454
	StdDev	0.02120	0.05373	0.07355	0.03546	0.03388
	Best	0.76582	0.95192	0.61111	0.71909	0.81250
	Worst	0.67722	0.72115	0.29630	0.53109	0.66560
GOA	Average	0.73122	0.49383	0.85449	0.64913	0.75555
	StdDev	0.02378	0.03809	0.02645	0.02879	0.01379
	Best	0.79114	0.57407	0.91346	0.71238	0.78793
	Worst	0.67722	0.42593	0.79808	0.58653	0.72685
GA	Average	0.71814	0.82372	0.51481	0.65030	0.76671
	StdDev	0.01757	0.03645	0.03587	0.01702	0.01242
	Best	0.75949	0.94231	0.55556	0.67792	0.78241
	Worst	0.68354	0.75962	0.40741	0.61048	0.73326
PSO	Average	0.72658	0.85096	0.48704	0.64126	0.76022
	StdDev	0.02768	0.05845	0.06018	0.02874	0.02382
	Best	0.77848	0.95192	0.61111	0.68990	0.79897
	Worst	0.64557	0.70192	0.38889	0.57689	0.71546
FPA	Average	0.72869	0.84231	0.50988	0.65336	0.76480
	StdDev	0.02307	0.04802	0.05728	0.02813	0.02171
	Best	0.76582	0.93269	0.64815	0.71050	0.80520
	Worst	0.68354	0.77885	0.38889	0.59377	0.72489
BAT	Average	0.72405	0.83654	0.50741	0.65086	0.76642
	StdDev	0.01840	0.03292	0.03248	0.01780	0.01025
	Best	0.75949	0.89423	0.59259	0.67779	0.78846
	Worst	0.68987	0.76923	0.42593	0.61715	0.74252
FF	Average	0.71730	0.51667	0.82147	0.65137	0.77276
	StdDev	0.01259	0.01709	0.01650	0.01288	0.00402
	Best	0.74051	0.55556	0.84615	0.68172	0.77902
	Worst	0.69620	0.48148	0.79808	0.62361	0.76086
MBO	Average	0.72932	0.83782	0.52037	0.65881	0.76113
	StdDev	0.02126	0.04101	0.05246	0.02611	0.02234
	Best	0.76582	0.89423	0.62963	0.70408	0.79790
	Worst	0.68354	0.75000	0.40741	0.59706	0.71599
BBO	Average	0.72911	0.83910	0.51728	0.65776	0.77369
	StdDev	0.01395	0.02976	0.04508	0.02235	0.00905
	Best	0.75316	0.91346	0.61111	0.69696	0.78775
	Worst	0.69620	0.78846	0.40741	0.60359	0.75036
GWO	Average	0.73543	0.82147	0.55496	0.67851	0.77645
	StdDev	0.01395	0.03155	0.04706	0.02451	0.00998
	Best	0.75424	0.90452	0.63257	0.68643	0.79846
	Worst	0.69857	0.79453	0.41564	0.61252	0.74049
FOA	Average	0.73424	0.83450	0.50476	0.64593	0.77227
	StdDev	0.01982	0.02998	0.04647	0.02649	0.01073
	Best	0.76476	0.90756	0.59786	0.69334	0.78246
	Worst	0.69175	0.78310	0.39461	0.60147	0.74378
WOA	Average	0.71352	0.80641	0.51345	0.63021	0.76503
	StdDev	0.01833	0.02842	0.04631	0.02653	0.02015
	Best	0.75836	0.89476	0.58302	0.68055	0.78101
	Worst	0.68705	0.77592	0.39305	0.59444	0.74212
SSA	Average	0.73017	0.82506	0.51139	0.64302	0.76218
	StdDev	0.02324	0.04275	0.04756	0.02602	0.02351
	Best	0.76603	0.89572	0.62641	0.70219	0.78595
	Worst	0.68231	0.75929	0.40453	0.59342	0.73890
BSOA	Average	0.72833	0.82683	0.52301	0.64981	0.76303
	StdDev	0.02256	0.04095	0.05036	0.02599	0.02264
	Best	0.76032	0.89210	0.61452	0.70619	0.79593
	Worst	0.68505	0.75410	0.41033	0.59513	0.72882

optimization experiment. In this experiment, besides the weights, the number of hidden units is also optimized. In a solution, the vector is extended with the maximum number of hidden units (Eq.18). By using binary encoding strategy, if the value is less than 0, the value will be 0, otherwise 1. The value 1 indicates to active units if the value is 0, the units are deactivated, and consequently, the weights and biases also become deactivated.

Table 14 presents the results of the average 30 independent runs. In the table, the three best solutions are presented for the breast cancer, diabetes, SAheart, and vertebral datasets, in the case of the Parkinson dataset, the five best solutions are presented considering the range of the possible hidden units are much bigger than in the other datasets.

TABLE 12. Vertebral dataset results.

Algorithms	Result	Accuracy	Specificity	Sensitivity	G-Mean	AUC
ABCQRBEST	Average	0.87063	0.89640	0.89753	0.90117	0.94892
	StdDev	0.01900	0.01013	0.05019	0.02961	0.00937
	Best	0.90721	0.91135	0.98712	0.94845	0.95733
	Worst	0.85510	0.88461	0.85128	0.86712	0.93152
ABC	Average	0.84371	0.81828	0.85422	0.83483	0.92189
	StdDev	0.03265	0.05522	0.05216	0.02786	0.01995
	Best	0.90566	0.93548	0.94667	0.87375	0.95441
	Worst	0.77358	0.67742	0.74667	0.76031	0.87484
GOA	Average	0.86321	0.88444	0.81183	0.84723	0.94060
	StdDev	0.00885	0.00948	0.02553	0.01307	0.00501
	Best	0.88679	0.90667	0.87097	0.87547	0.95140
	Worst	0.84906	0.86667	0.77419	0.82540	0.93204
GA	Average	0.85786	0.82151	0.87289	0.84664	0.93508
	StdDev	0.01049	0.02198	0.01706	0.01026	0.00465
	Best	0.88679	0.83871	0.92000	0.87203	0.94495
	Worst	0.83962	0.77419	0.84000	0.82540	0.92602
PSO	Average	0.85031	0.78925	0.87556	0.82942	0.91911
	StdDev	0.03349	0.07023	0.05628	0.03254	0.01703
	Best	0.90566	0.90323	0.97333	0.87203	0.94108
	Worst	0.77358	0.61290	0.76000	0.73441	0.87570
FPA	Average	0.86730	0.77742	0.90444	0.83816	0.92433
	StdDev	0.01506	0.03425	0.02351	0.01676	0.00836
	Best	0.88679	0.83871	0.96000	0.86559	0.94065
	Worst	0.83962	0.70968	0.86667	0.80215	0.90538
BAT	Average	0.83553	0.81075	0.84578	0.82617	0.91209
	StdDev	0.05471	0.05272	0.08269	0.04071	0.04299
	Best	0.89623	0.90323	0.93333	0.87841	0.94796
	Worst	0.68868	0.67742	0.62667	0.68533	0.77247
FF	Average	0.86384	0.87911	0.82688	0.85250	0.94072
	StdDev	0.00810	0.01259	0.01794	0.00860	0.00434
	Best	0.87736	0.90667	0.83871	0.86559	0.94968
	Worst	0.84906	0.85333	0.77419	0.82540	0.92946
MBO	Average	0.85660	0.78387	0.88667	0.83203	0.92346
	StdDev	0.02796	0.06460	0.05083	0.02750	0.01766
	Best	0.90566	0.87097	0.98667	0.89515	0.94882
	Worst	0.76415	0.64516	0.73333	0.77598	0.87957
BBO	Average	0.86730	0.82473	0.88489	0.85403	0.93875
	StdDev	0.01106	0.02896	0.01836	0.01285	0.00660
	Best	0.88679	0.87097	0.93333	0.87547	0.95097
	Worst	0.84906	0.74194	0.84000	0.83163	0.92645
GWO	Average	0.85421	0.78623	0.87656	0.83341	0.92013
	StdDev	0.03148	0.06024	0.04631	0.02234	0.01514
	Best	0.90366	0.90125	0.97532	0.87301	0.93999
	Worst	0.79353	0.63295	0.75903	0.75443	0.87681
FOA	Average	0.85332	0.81363	0.87483	0.84093	0.92979
	StdDev	0.01231	0.02756	0.01746	0.01375	0.00761
	Best	0.88971	0.88095	0.94313	0.88543	0.94597
	Worst	0.83706	0.73995	0.8351	0.83062	0.92743
WOA	Average	0.83651	0.81174	0.84372	0.82736	0.91351
	StdDev	0.05591	0.05163	0.08279	0.04043	0.04354
	Best	0.89425	0.90221	0.93243	0.87745	0.94694
	Worst	0.68651	0.67643	0.62562	0.68310	0.77628
SSA	Average	0.85685	0.82243	0.87351	0.84436	0.93217
	StdDev	0.01053	0.02152	0.01936	0.01046	0.00563
	Best	0.88571	0.83601	0.92005	0.87132	0.94293
	Worst	0.83852	0.77350	0.84301	0.82390	0.92732
BSOA	Average	0.83753	0.81262	0.84561	0.82625	0.91232
	StdDev	0.05583	0.05270	0.08095	0.04153	0.04230
	Best	0.89492	0.90265	0.93371	0.87521	0.94704
	Worst	0.68543	0.67546	0.62573	0.68251	0.77639

TABLE 13. Best result summary.

Algorithms	Average	StdDev	Best	Worst	TOTAL
ABCQRBEST	14	2	19	11	46
ABC	0	1	5	0	6
GOA	3	0	3	2	8
GA	0	0	2	1	3
PSO	2	0	3	0	5
FPA	2	0	1	1	4
BAT	0	1	1	1	3
FF	2	17	1	5	25
MBO	0	0	3	0	3
BBO	0	3	1	3	7
GWO	0	1	1	0	2
FOA	0	0	1	0	1
WOA	0	0	1	0	1
SSA	0	0	2	0	2
BSOA	0	0	1	0	1

Based on the obtained results, it can be concluded that with fewer hidden units, it is possible to achieve the same or even better accuracy, and on the other hand, the computation time is reduced significantly.

TABLE 14. Hidden unit number optimization.

Dataset	Rank	Hidden units	Accuracy	Specificity	Sensitivity	G-Mean	AUC
Breast cancer	1	14	0.98540	0.98876	0.97916	0.98395	0.98396
	2	19	0.98534	0.97752	1.00000	0.98870	0.98000
	3	15	0.97810	0.98947	0.95238	0.97075	0.97738
Diabetes	1	10	0.80749	0.88423	0.91969	0.90178	0.86677
	2	15	0.80625	0.83848	0.87368	0.85590	0.86988
	3	13	0.80575	0.84932	0.83788	0.84358	0.85897
Parkinson	1	8	0.97080	0.97115	0.96969	0.97042	0.95224
	2	15	0.96350	0.96116	0.97058	0.96586	0.94094
	3	14	0.95620	0.93750	1.00000	0.96824	0.93617
	4	26	0.94890	0.92783	1.00000	0.96324	0.92553
	5	6	0.89743	0.50000	0.89743	0.66986	0.50000
Saheart	1	12	0.79362	0.81698	0.87167	0.84388	0.81990
	2	19	0.79254	0.84435	0.86410	0.85416	0.79969
	3	9	0.78723	0.77391	0.83226	0.80255	0.70256
Vertebral	1	11	0.90872	0.89565	0.84074	0.86776	0.86398
	2	9	0.90856	0.86923	0.88205	0.87561	0.82549
	3	5	0.90762	0.88462	0.82051	0.85196	0.78842

VI. CONCLUSION

In this work, an improved ABC algorithm is proposed for weight connection optimization and hidden unit number optimization in neural networks. The proposed method improves the exploitation capability of the basic ABC algorithm, by incorporating quasi-reflection-based learning and random current best solution mechanisms. The algorithm is tested on the recent challenging CEC 2017 benchmark function suite to test the exploration and exploitation capability and solution quality. The obtained results are compared to the original ABC and other recent metaheuristics approaches, and the statistical results shows the robustness of ABCQRBEST, which results are significantly better than the result of other approaches.

The proposed ABCQRBEST is employed for weight connection and hidden unit number optimization in neural networks. For evaluating the performance, the simulations are conducted on five well-known medical datasets. The obtained statistical results are compared to similar metaheuristic based approaches. The proposed method outperformed the other, current metaheuristic based methods.

The limitations of the proposed algorithm are as follows. First, it is necessary to put additional effort to set up the algorithm, in terms of control parameters, for a particular problem that is being solved, and it is done empirically, by trial and error. The second drawback of the algorithm is that it requires more calculations in each iteration, due to the generation of the quasi reflexive learning population. This means $NP * FFE$ more evaluations of the fitness function in each iteration. However, as shown in experiments, when the $maxFFE$ is taken as the termination condition, this drawback is only conditional.

Based on the findings, it can concluded that ABCQRBEST is very promising and competitive over the current approaches in neural network weight optimization and hidden unit optimization. In the future research, plan is to adapt and to test devised method for other datasets, as well as to combine it with other machine learning algorithms.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[2] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," in *Proc. IJCAI*, vol. 89, 1989, pp. 762–767.

[3] W. Li, "Improving particle swarm optimization based on neighborhood and historical memory for training multi-layer perceptron," *Information*, vol. 9, no. 1, p. 16, Jan. 2018.

[4] H. Hakli and M. S. Kiran, "An improved artificial bee colony algorithm for balancing local and global search behaviors in continuous optimization," *Int. J. Mach. Learn. Cybern.*, vol. 11, pp. 1–26, Feb. 2020.

[5] V. K. Ojha, A. Abraham, and V. Snášel, "Metaheuristic design of feedforward neural networks: A review of two decades of research," *Eng. Appl. Artif. Intell.*, vol. 60, pp. 97–116, Apr. 2017.

[6] V. K. Ojha, A. Abraham, and V. Snášel, "Simultaneous optimization of neural network weights and active nodes using metaheuristics," in *Proc. 14th Int. Conf. Hybrid Intell. Syst.*, Dec. 2014, pp. 248–253.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw. (ICNN)*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.

[8] M. Kumar and S. C. Sharma, "PSO-COGEN: Cost and energy efficient scheduling in cloud environment with deadline constraint," *Sustain. Comput., Informat. Syst.*, vol. 19, pp. 147–164, Sep. 2018.

[9] M. Abdel-Basset, N. Moustafa, R. Mohamed, O. M. Elkomy, and M. Abouhawwash, "Multi-objective task scheduling approach for fog computing," *IEEE Access*, vol. 9, pp. 126988–127009, 2021.

[10] M. Dorigo and M. Birattari, *Ant Colony Optimization*. Berlin, Germany: Springer-Verlag, 2010.

[11] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 8, pp. 687–697, Jan. 2008.

[12] N. Bacanin and M. Tuba, "Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators," *Stud. Informat. Control*, vol. 21, no. 2, pp. 137–146, Jun. 2012.

[13] M. Tuba and N. Bacanin, "Artificial bee colony algorithm hybridized with firefly algorithm for cardinality constrained mean-variance portfolio selection problem," *Appl. Math. Inf. Sci.*, vol. 8, no. 6, pp. 2831–2844, Nov. 2014.

[14] V. R. Kulkarni, V. Desai, and R. V. Kulkarni, "Multistage localization in wireless sensor networks using artificial bee colony algorithm," in *Proc. IEEE Symp. Series Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.

[15] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, O. Watanabe and T. Zeugmann, Eds. Berlin, Germany: Springer, 2009, pp. 169–178.

[16] M. Zivkovic, N. Bacanin, E. Tuba, I. Strumberger, T. Bezdan, and M. Tuba, "Wireless sensor networks life time optimization based on the improved firefly algorithm," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, Jun. 2020, pp. 1176–1181.

[17] X.-S. Yang and A. H. Gandomi, "Bat algorithm: A novel approach for global engineering optimization," *Eng. Comput.*, vol. 29, no. 5, pp. 464–483, 2012.

[18] T. Bezdan, M. Zivkovic, E. Tuba, I. Strumberger, N. Bacanin, and M. Tuba, "Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm," in *Proc. Int. Conf. Intell. Fuzzy Syst. Cham, Switzerland: Springer*, 2020, pp. 718–725.

[19] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, Feb. 2016.

[20] N. Bacanin, E. Tuba, M. Zivkovic, I. Strumberger, and M. Tuba, "Whale optimization algorithm with exploratory move for wireless sensor networks localization," in *Proc. Int. Conf. Hybrid Intell. Syst. Cham, Switzerland: Springer*, 2019, pp. 328–338.

[21] G.-G. Wang, S. Deb, and L. S. dos Coelho, "Elephant herding optimization," in *Proc. 3rd Int. Symp. Comput. Bus. Intell. (ISCBI)*, Dec. 2015, pp. 1–5.

[22] I. Strumberger, E. Tuba, M. Zivkovic, N. Bacanin, M. Beko, and M. Tuba, "Dynamic search tree growth algorithm for global optimization," in *Technological Innovation for Industry and Service Systems*, L. M. Camarinha-Matos, R. Almeida, and J. Oliveira, Eds. Cham, Switzerland: Springer, 2019, pp. 143–153.

[23] G.-G. Wang, "Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Comput.*, vol. 10, no. 2, pp. 151–164, Sep. 2016.

[24] G.-G. Wang, "Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Comput.*, vol. 10, pp. 151–164, Jun. 2018.

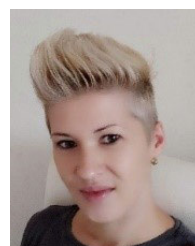
[25] I. Strumberger, M. Sarac, D. Markovic, and N. Bacanin, "Moth search algorithm for drone placement problem," *Int. J. Comput.*, vol. 3, pp. 1–6, Apr. 2018.

- [26] E. Tuba, I. Strumberger, D. Zivkovic, N. Bacanin, and M. Tuba, "Mobile robot path planning by improved brain storm optimization algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8.
- [27] I. Strumberger, M. Minovic, M. Tuba, and N. Bacanin, "Performance of elephant herding optimization and tree growth algorithm adapted for node localization in wireless sensor networks," *Sensors*, vol. 19, no. 11, p. 2515, Jun. 2019.
- [28] M. Zivkovic, N. Bacanin, T. Zivkovic, I. Strumberger, E. Tuba, and M. Tuba, "Enhanced grey wolf algorithm for energy efficient wireless sensor networks," in *Proc. Zooming Innov. Consum. Technol. Conf. (ZINC)*, May 2020, pp. 87–92.
- [29] I. Strumberger, N. Bacanin, M. Tuba, and E. Tuba, "Resource scheduling in cloud computing based on a hybridized whale optimization algorithm," *Appl. Sci.*, vol. 9, no. 22, p. 4893, Nov. 2019.
- [30] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, M. Tuba, and M. Zivkovic, "Task scheduling in cloud computing environment by grey wolf optimizer," in *Proc. 27th Telecommun. Forum (TELFOR)*, Nov. 2019, pp. 1–4.
- [31] M. Zivkovic, N. Bacanin, K. Venkatachalam, A. Nayyar, A. Djordjevic, I. Strumberger, and F. Al-Turjman, "COVID-19 cases prediction by using hybrid machine learning and beetle antennae search approach," *Sustain. Cities Soc.*, vol. 66, Mar. 2021, Art. no. 102669.
- [32] M. Zivkovic, K. Venkatachalam, N. Bacanin, A. Djordjevic, M. Antonijevic, I. Strumberger, and T. A. Rashid, "Hybrid genetic algorithm and machine learning method for COVID-19 cases prediction," in *Proc. Int. Conf. Sustain. Expert Syst. (ICSES)*, vol. 176. Singapore: Springer, 2021, p. 169.
- [33] N. Bacanin, A. Petrovic, M. Zivkovic, T. Bezdan, and M. Antonijevic, "Feature selection in machine learning by hybrid sine cosine metaheuristics," in *Proc. Int. Conf. Adv. Comput. Data Sci.* Cham, Switzerland: Springer, 2021, pp. 604–616.
- [34] N. Bacanin, A. Petrovic, M. Zivkovic, T. Bezdan, and A. Chhabra, "Enhanced salp swarm algorithm for feature selection," in *Proc. Int. Conf. Intell. Fuzzy Syst.* Cham, Switzerland: Springer, 2021, pp. 483–491.
- [35] N. Bacanin, N. Vukobrat, M. Zivkovic, T. Bezdan, and I. Strumberger, "Improved Harris hawks optimization adapted for artificial neural network training," in *Proc. Int. Conf. Intell. Fuzzy Syst.* Cham, Switzerland: Springer, 2021, pp. 281–289.
- [36] J. Basha, N. Bacanin, N. Vukobrat, M. Zivkovic, K. Venkatachalam, S. Hubálovský, and P. Trojovský, "Chaotic Harris hawks optimization with quasi-reflection-based learning: An application to enhance CNN design," *Sensors*, vol. 21, no. 19, p. 6654, Oct. 2021.
- [37] N. Bacanin, K. Alhazmi, M. Zivkovic, K. Venkatachalam, T. Bezdan, and J. Nebhen, "Training multi-layer perceptron with enhanced brain storm optimization metaheuristics," *Comput., Mater. Continua*, vol. 70, no. 2, pp. 4199–4215, 2022.
- [38] N. Bacanin, R. Stoean, M. Zivkovic, A. Petrovic, T. A. Rashid, and T. Bezdan, "Performance of a novel chaotic firefly algorithm with enhanced exploration for tackling global optimization problems: Application for dropout regularization," *Mathematics*, vol. 9, no. 21, p. 2705, Oct. 2021.
- [39] Z. Tian, Y. Ren, and G. Wang, "Short-term wind speed prediction based on improved PSO algorithm optimized EM-ELM," *Energy Sour., A, Recovery, Utilization, Environ. Effects*, vol. 41, no. 1, pp. 26–46, Jan. 2019.
- [40] T. Bezdan, C. Stoean, A. A. Naamany, N. Bacanin, T. A. Rashid, M. Zivkovic, and K. Venkatachalam, "Hybrid fruit-fly optimization algorithm with K-means for text document clustering," *Mathematics*, vol. 9, no. 16, p. 1929, Aug. 2021.
- [41] T. Bezdan, M. Zivkovic, E. Tuba, I. Strumberger, N. Bacanin, and M. Tuba, "Glioma brain tumor grade classification from MRI using convolutional neural networks designed by modified FA," in *Proc. Int. Conf. Intell. Fuzzy Syst.* Cham, Switzerland: Springer, 2020, pp. 955–963.
- [42] N. Sulaiman, J. Mohamad-Saleh, and A. G. Abro, "A hybrid algorithm of ABC variant and enhanced EGS local search technique for enhanced optimization performance," *Eng. Appl. Artif. Intell.*, vol. 74, pp. 10–22, Sep. 2018.
- [43] Z. Tian, "Echo state network based on improved fruit fly optimization algorithm for chaotic time series prediction," *J. Ambient Intell. Humanized Comput.*, pp. 1–20, Apr. 2020, doi: [10.1007/s12652-020-01920-4](https://doi.org/10.1007/s12652-020-01920-4).
- [44] Z. Tian, "Modes decomposition forecasting approach for ultra-short-term wind speed," *Appl. Soft Comput.*, vol. 105, Jul. 2021, Art. no. 107303.
- [45] Z. Tian and H. Chen, "A novel decomposition-ensemble prediction model for ultra-short-term wind speed," *Energy Convers. Manage.*, vol. 248, Nov. 2021, Art. no. 114775.
- [46] Z. Tian and H. Chen, "Multi-step short-term wind speed prediction based on integrated multi-model fusion," *Appl. Energy*, vol. 298, Sep. 2021, Art. no. 117248.
- [47] Z. Tian, H. Li, and F. Li, "A combination forecasting model of wind speed based on decomposition," *Energy Rep.*, vol. 7, pp. 1217–1233, Nov. 2021.
- [48] A. A. Heidari, H. Faris, I. Aljarah, and S. Mirjalili, "An efficient hybrid multilayer perceptron neural network with grasshopper optimization," *Soft Comput.*, vol. 23, no. 17, pp. 7941–7958, 2019.
- [49] L. Haghnegahdar and Y. Wang, "A whale optimization algorithm-trained artificial neural network for smart grid cyber intrusion detection," *Neural Comput. Appl.*, vol. 32, no. 13, pp. 9427–9441, Jul. 2020.
- [50] I. Aljarah, H. Faris, and S. Mirjalili, "Optimizing connection weights in neural networks using the whale optimization algorithm," *Soft Comput.*, vol. 22, no. 1, pp. 1–15, 2018.
- [51] U. Agrawal, J. Arora, R. Singh, D. Gupta, A. Khanna, and A. Khamparia, "Hybrid wolf-bat algorithm for optimization of connection weights in multi-layer perceptron," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 16, no. 1s, pp. 1–20, Apr. 2020.
- [52] Z. Gao, Y. Li, Y. Yang, X. Wang, N. Dong, and H.-D. Chiang, "A GPSO-optimized convolutional neural networks for EEG-based emotion recognition," *Neurocomputing*, vol. 380, pp. 225–235, Mar. 2020.
- [53] A. Darwish, D. Ezzat, and A. E. Hassanien, "An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis," *Swarm Evol. Comput.*, vol. 52, Feb. 2020, Art. no. 100616.
- [54] G. H. de Rosa, J. P. Papa, and X.-S. Yang, "Handling dropout probability estimation in convolution neural networks using meta-heuristics," *Soft Comput.*, vol. 22, no. 18, pp. 6147–6156, Sep. 2018.
- [55] Y. Wang, H. Zhang, and G. Zhang, "CPSO-CNN: An efficient PSO-based algorithm for fine-tuning hyper-parameters of convolutional neural networks," *Swarm Evol. Comput.*, vol. 49, pp. 114–123, Sep. 2019.
- [56] T. Yamasaki, T. Honma, and K. Aizawa, "Efficient optimization of convolutional neural networks using particle swarm optimization," in *Proc. IEEE 3rd Int. Conf. Multimedia Big Data*, Apr. 2017, pp. 70–73.
- [57] A. K. Anaraki, M. Ayati, and F. Kazemi, "Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms," *Biocybernetics Biomed. Eng.*, vol. 39, no. 1, pp. 63–74, Jan./Mar. 2019.
- [58] C. Fernando, D. Banarse, M. Reynolds, F. Besse, D. Pfau, M. Jaderberg, M. Lanctot, and D. Wierstra, "Convolution by evolution: Differentiable pattern producing networks," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 109–116.
- [59] J. Davison. (2017). *DEvol: Automated Deep Neural Network Design. Via Genetic Programming*. [Online]. Available: <https://github.com/joedddav/devol>
- [60] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 2121–2159, 2011.
- [61] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*.
- [62] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [63] A. Y. Ng, "Feature selection, L_1 vs. L_2 regularization, and rotational invariance," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2004, p. 78.
- [64] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jun. 2014.
- [65] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1058–1066.
- [66] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., Lille, France, vol. 37, Jul. 2015, pp. 448–456.
- [67] A. Krogh, "What are artificial neural networks?" *Nature Biotechnol.*, vol. 26, no. 2, pp. 195–197, 2008.
- [68] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.
- [69] Y.-C. Hu, "Nonadditive similarity-based single-layer perceptron for multi-criteria collaborative filtering," *Neurocomputing*, vol. 129, pp. 306–314, Apr. 2014.

- [70] H. Faris, I. Aljarah, N. Al-Madi, and S. Mirjalili, "Optimizing the learning process of feedforward neural networks using lightning search algorithm," *Int. J. Artif. Intell. Tools*, vol. 25, no. 6, Dec. 2016, Art. no. 1650033.
- [71] M. Subotic and M. Tuba, "Parallelized multiple swarm artificial bee colony algorithm (MS-ABC) for global optimization," *Stud. Informat. Control*, vol. 23, no. 1, pp. 117–126, Mar. 2014.
- [72] D. Karaboga and B. Akay, "A modified artificial bee colony (ABC) algorithm for constrained optimization problems," *Appl. Soft Comput.*, vol. 11, no. 3, pp. 3021–3031, Apr. 2011.
- [73] I. Brajevic and M. Tuba, "An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems," *J. Intell. Manuf.*, vol. 24, no. 4, pp. 729–740, Aug. 2013.
- [74] N. Bacanin and M. Tuba, "Firefly algorithm for cardinality constrained mean-variance portfolio optimization problem with entropy diversity constraint," *Sci. World J., Special Issue Comput. Intell. Metaheuristic Algorithms With Appl.*, vol. 2014, May 2014, Art. no. 721521.
- [75] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Quasi-oppositional differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 2229–2236.
- [76] N. H. Awad, M. Z. Ali, J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization," Zhengzhou Univ., Zhengzhou, China, Tech. Rep., 2016.
- [77] S. Gupta and K. Deep, "Improved sine cosine algorithm with crossover scheme for global optimization," *Knowl.-Based Syst.*, vol. 165, pp. 374–406, Feb. 2019.
- [78] A. G. Hussien and M. Amin, "A self-adaptive Harris hawks optimization algorithm with opposition-based learning and chaotic local search strategy for global optimization and feature selection," *Int. J. Mach. Learn. Cybern.*, pp. 1–28, Apr. 2021, doi: 10.1007/s13042-021-01326-4.
- [79] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.
- [80] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.
- [81] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, and I. Aljarah, "Grasshopper optimization algorithm for multi-objective optimization problems," *Appl. Intell.*, vol. 48, no. 4, pp. 805–820, 2018.
- [82] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
- [83] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.
- [84] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: A nature-inspired algorithm for global optimization," *Neural Comput. Appl.*, vol. 27, no. 2, pp. 495–513, Feb. 2016.
- [85] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, Nov. 1995, pp. 1942–1948.
- [86] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.
- [87] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Statist. Assoc.*, vol. 32, no. 200, pp. 675–701, Dec. 1937.
- [88] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Statist.*, vol. 11, pp. 86–92, Mar. 1940.
- [89] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the fbietkan statistic," *Commun. Statist.-Theory Methods*, vol. 9, no. 6, pp. 571–595, 1980.
- [90] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. London, U.K.: Chapman & Hall, 2020.
- [91] D. Dua and C. Graff, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, 2019. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [92] O. L. Mangasarian and W. H. Wolberg, "Cancer diagnosis via linear programming," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, 1990.
- [93] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proc. Nat. Acad. Sci. USA*, vol. 87, no. 23, pp. 9193–9196, Dec. 1990.
- [94] M. A. Little, P. E. McSharry, S. J. Roberts, D. A. E. Costello, and I. M. Moroz, "Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection," *BioMed. Eng. OnLine*, vol. 6, no. 1, p. 23, 2007.
- [95] J. Rossouw, J. Du Plessis, A. Benadé, P. Jordaan, J. Kotze, P. Jooste, and J. Ferreira, "Coronary risk factor screening in three rural communities. The CORIS baseline study," *South Afr. Med. J.=Suid-Afrikaanse Tydskrif vir Geneeskunde*, vol. 64, no. 12, p. 430, 1983.
- [96] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Proc. Int. Conf. Unconventional Comput. Natural Comput.* Berlin, Germany: Springer, 2012, pp. 240–249.
- [97] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Comput. Appl.*, vol. 31, pp. 1–20, May 2015.
- [98] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.
- [99] W.-T. Pan, "A new fruit fly optimization algorithm: Taking the financial distress model as an example," *Knowl.-Based Syst.*, vol. 26, pp. 69–74, Feb. 2012.
- [100] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.
- [101] Y. Shi, "Brain storm optimization algorithm," in *Proc. Int. Conf. Swarm Intell.* Berlin, Germany: Springer, 2011, pp. 303–309.



NEBOJSA BACANIN received the Ph.D. degree from the Faculty of Mathematics, University of Belgrade, in 2015, (study program computer science, average grade 10,00). He started university career in Serbia 15 years ago with the Graduate School of Computer Science, Belgrade. He is currently working as an Associate Professor and as a Vice-Rector of scientific research with Singidunum University, Belgrade, Serbia. He is involved in scientific research in the field of computer science and his specialty include stochastic optimization algorithms, swarm intelligence, soft-computing and optimization and modeling, and artificial intelligence algorithms, swarm intelligence, machine learning, image processing, and cloud and distributed computing. He has published more than 170 scientific papers in high quality journals and international conferences indexed in Clarivate Analytics JCR, Scopus, WoS, IEEEExplore, and other scientific databases, and in Springer and *Procedia Computer Science* book chapters. He has also published four books from the domains of cloud computing, web programming, and advanced java spring programming. He is also a member of numerous editorial boards, scientific and advisory committees of international conferences and journals. He is also a Regular Reviewer for international journals with high Clarivate Analytics and WoS impact factor. He actively participates in one national and one international projects from the domain of computer science. He has also been included in the prestigious Stanford University single year list with 2% best world researchers, in 2020 and 2021.



TIMEA BEZDÁN received the B.S. degree in software and data engineering from Singidunum University, Belgrade, Serbia, in 2020, where she is currently pursuing the Ph.D. degree in computer science. She is currently working as a Teaching Assistant at the Technical Faculty and Faculty of Informatics and Computing, Singidunum University. She has published more than 30 scientific papers in high-quality journals and international conferences in these areas. Her research interests include artificial intelligence, machine learning, optimization, swarm intelligence, and cloud computing.



K. VENKATACHALAM received the bachelor's degree in information technology, in 2005, the master's degree in computer science and engineering, in 2008, and the Ph.D. degree in computer science and engineering, in 2018. He has more than 13 years of academic experience and currently working as a Senior Research with the Department of Applied Cybernetics, Faculty of Science, University of Hradec Kralove, Hradec Kralove, Czech Republic. He has published several articles in peer-reviewed journals. His research interests include data mining, web services, semantic web services, distributed computing, and cloud computing. He is a Sun certified SCJP professional and has obtained Brain Bench certification in various disciplines. He has organized several workshops on J2ME, advanced java programming, web services, enterprise computing, web technology, and wireless sensor network in his institution and has presented papers in web services at national and international conferences. He has guided a number of research-oriented and application-oriented projects organized by well-known companies, such as IBM. He has delivered more than 20 guest lectures in reputed engineering colleges on various topics.



MIODRAG ZIVKOVIC received the Ph.D. degree from the School of Electrical Engineering, University of Belgrade, in 2014, (study program software engineering, average grade 9,90). He started university career in Serbia six years ago with Singidunum University, Belgrade, Serbia, where he is currently working as an Associate Professor with the Faculty of Informatics and Computing. He is involved in scientific research in the field of computer science and his specialty includes object-oriented programming, mobile applications programming, software testing, stochastic optimization algorithms, swarm intelligence, human-computer interaction, and artificial intelligence algorithms. He has published more than 70 scientific papers in high quality journals and international conferences indexed in Clarivate Analytics JCR, Scopus, WoS, IEEEExplore, and other scientific databases, and in Springer Lecture Notes in computer science. He has also published three books in domains of software testing, mobile applications development, and advanced java spring programming. He is also a Regular Reviewer for international journals with high Clarivate Analytics and WoS impact factor.



IVANA STRUMBERGER received the Ph.D. degree from Singidunum University, Belgrade, Serbia, in the domain of computer science (average grade: 9,93), in 2020. She started her university career as a Teaching Assistant with the Faculty of Computer Science, Belgrade, in 2013. She is currently working as an Assistant Professor with the Faculty of Informatics and Computing, Singidunum University. She conducts research in the domain of computer science and her specialty includes swarm intelligence, machine learning, optimization and modeling, cloud computing, computer networks, and distributed computing. She has published more than 70 scientific papers in high quality journals and international conferences indexed in Clarivate Analytics JCR, Scopus, WoS, and IEEEExplore. She has also published 15 book chapters in Springer Lecture Notes in computer science series and two books from the domain of cloud computing. She is a Regular Reviewer of many international state-of-the-art journals with high Clarivate Analytics and WoS impact factor.



MOHAMED ABOUHAUWASH received the B.Sc. and M.Sc. degrees in statistics and computer science from Mansoura University, Mansoura, Egypt, in 2005 and 2011, respectively, and the Ph.D. degrees in statistics and computer science, in a channel program from Michigan State University, East Lansing, MI, USA, and Mansoura University, in 2015. He is currently with Computational Mathematics, Science, and Engineering (CMSE), Biomedical Engineering (BME) and Radiology, Institute for Quantitative Health Science & Engineering (IQ), Michigan State University. In 2018, he was a Visiting Scholar with the Department of Mathematics and Statistics, Faculty of Science, Thompson Rivers University, Kamloops, BC, Canada. He is currently an Associate Professor with the Department of Mathematics, Faculty of Science, Mansoura University. His current research interests include evolutionary algorithms, machine learning, image reconstruction, and mathematical optimization. He was a recipient of the best master's and Ph.D. thesis awards from Mansoura University, in 2012 and 2018, respectively.



ABEER B. AHMED received the Ph.D. degree with entrepreneurial leadership experience in stock market and computer science domains. She is currently the Vice Dean for training and community services, College of Computing and Information Technology, Arab Academy for Science, Technology and Maritime. She is Passionate for understanding and positioning technology and products, educating customers and working in consultancy driven business environments. She is also a Research Professional and a Technical Analyst with prime specialization in the use of intelligent methods to discover hidden patterns within stock data. She is having project management skills for research and software development of stock market related product suits. She is a Founder and the Managing Director of Middle East Sentiment Consultant Inc., (MESCI). The company's clientele include major investment corporate in Egypt, Saudi Arabia, and United Arab Emirates.

...