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

NEBOJSA BACANIN1, TIMEA BEZDAN1 K. VENKATACHALAM2, MIODRAG ZIVKOVIC1, IVANA STRUMBERGER1, MOHAMED ABOUHAWWASH3,4, and ABEER AHMED5

1Department of Computer Science, Singidunum University, 11000 Belgrade, Serbia, (email: nbacanin@singidunum.ac.rs, tbdezdan@singidunum.ac.rs, mzivkovic@singidunum.ac.rs, istrumberger@singidunum.ac.rs)
2Department of Applied Cybernetics, Faculty of Science, University of Hradec Králové, Hradec Králové, Czech Republic, (email:venkatachalam.k@ieee.org)
3Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt, (email:saleh1284@mans.edu.eg).
4Department of Computational Mathematics, Science, and Engineering (CMSE), Michigan State University, East Lansing, MI, 48824 USA,(email: abouhaww@msu.edu)
5Computer engineering department, College of computing and information technology, Arab Academy for science technology and maritim, Cairo, Egypt,(email: abadredin@aast.edu).

Corresponding author: Nebojsa Bacanin (email: 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-refection-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-refection-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. 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 backpropagation [1] are used most commonly for neural network optimization, which are deterministic approaches, and they have a disadvantage of local optimia 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], [4], [5], [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 [29], [18], [30], COVID-19 cases prediction [31], [32], feature selection problem [33], [34], ANN and CNN training optimization [35], [36], [37], [38], [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], [36], [37], [38], time series prediction [43], [44], [31], to the wind speed forecasting [45], [46], [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 neurorevolution 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 were 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 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], [61], [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 con-
neurons 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):

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

here, \(n\) stands for the number of input values, \(I_i\) represents the input value \(i\), \(\omega_{ij}\) is the connection weight, and finally, \(\beta_j\) denotes the bias term.

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):

$$f_j(x) = \frac{1}{1 + e^{-S_j}}$$  \hspace{1cm} (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:

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

$$x_{i,j} = lb_j + rand(0,1) \ast (ub_j - lb_j),$$  \hspace{1cm} (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):

$$v_{i,j} = \begin{cases} x_{i,j} + \phi \ast (x_{i,j} - x_{k,j}), & R_j < MR \\ x_{i,j}, \text{otherwise} \end{cases}$$  \hspace{1cm} (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 \(k\), \(\phi\) 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):
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 \( \text{limit} \) 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 exhibit 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.

\[
x_{\text{best}} = \min(x_{\text{best}}) + r \times (\max(x_{\text{best}}) - \min(x_{\text{best}})),
\]

where the best solution is denoted by \( x_{\text{best}} \), the minimum value of the best solution’s element is denoted by \( \min(x_{\text{best}}) \), \( \max(x_{\text{best}}) \) 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:

\[
x^o_j = lb_j + ub_j - x_j,
\]

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:

\[
x^{qo}_j = \frac{lb_j + ub_j}{2},
\]

where the mean of the lower bound and upper bound is calculated by \( \frac{lb_j + ub_j}{2} \), \( \text{rnd}(\frac{lb_j + ub_j}{2}, x^o_j) \) generates random number from uniform distribution in range \( \frac{lb_j + ub_j}{2}, x^o_j \).

The quasi-reflected component \( x^{qr}_j \) is defined as reflection of \( x^{qo}_j \) and calculated as:

\[
x^{qr}_j = \frac{lb_j + ub_j}{2},
\]

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

**Algorithm 2** Pseudo-code of the proposed ABCQRBEST algorithm

```plaintext
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
        $x_{i,j}^{qr} = x_{i,j} + \left(\frac{lb_j + ub_j}{2} - x_{i,j}\right) \cdot rand$
      else
        $x_{i,j}^{qr} = \frac{lb_j + ub_j}{2} + \left(x_{i,j} - \frac{lb_j + ub_j}{2}\right) \cdot rand$
      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
```

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

**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 $FFE$s [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 $FFE$s 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 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 trig-
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 $\alpha$, 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 ($NP \times NP \times T$).

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 ($NP$ in this case to 250). The configuration of the control parameters for the opposing methods can be found in [78].

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.
WOA has shown the best stability. For benchmark F10, QRBEST has shown better stability than DE, while the F24 and F27. Considering the F18, the proposed ABCQRBEST outperformed all other approaches on benchmarks F18, on benchmarks F10 and F16, while the DE method than DE.

Additionally, for F15, ABCQRBEST has better stability than DE.

It can also be noted that ABCQRBEST was outperformed by IHLO 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).

### Table 2: CEC 2017 comparative analysis results.

| Algorithm | Mean | STD | Mean | STD | Mean | STD |
|-----------|------|-----|------|-----|------|-----|
| IHLO      | 2.47E+3 | 2.26E+2 | 17.82 | 2.54E+2 | 18.246 | 7.56E+2 |
| DE        | 3.20E+3 | 2.72E+2 | 17.713 | 2.92E+2 | 6.59E+2 | 17.243 |
| GOA       | 3.13E+3 | 2.95E+2 | 17.313 | 2.92E+2 | 6.59E+2 | 17.243 |
| MVO       | 2.95E+3 | 2.55E+2 | 21.875 | 2.40E+2 | 6.42E+2 | 17.243 |
| WOA       | 3.13E+3 | 2.55E+2 | 21.875 | 2.40E+2 | 6.42E+2 | 17.243 |
| ABC       | 2.31E+3 | 1.05E+2 | 13.162 | 7.56E+2 | 17.243 | 7.56E+2 |
| ABO       | 1.80E+3 | 1.75E+2 | 13.162 | 7.56E+2 | 17.243 | 7.56E+2 |
| IHLO      | 2.47E+3 | 2.26E+2 | 17.82 | 2.54E+2 | 18.246 | 7.56E+2 |
| DE        | 3.20E+3 | 2.72E+2 | 17.713 | 2.92E+2 | 6.59E+2 | 17.243 |
| GOA       | 3.13E+3 | 2.95E+2 | 17.313 | 2.92E+2 | 6.59E+2 | 17.243 |
| MVO       | 2.95E+3 | 2.55E+2 | 21.875 | 2.40E+2 | 6.42E+2 | 17.243 |
| WOA       | 3.13E+3 | 2.55E+2 | 21.875 | 2.40E+2 | 6.42E+2 | 17.243 |
| ABC       | 2.31E+3 | 1.05E+2 | 13.162 | 7.56E+2 | 17.243 | 7.56E+2 |
| ABO       | 1.80E+3 | 1.75E+2 | 13.162 | 7.56E+2 | 17.243 | 7.56E+2 |

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
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.

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

| Function | IHHO | HHO | DE | COA | GWO | MFO | MVO | PSO | WOA | SCA | FA | ABCQRBEST |
|----------|------|-----|----|-----|-----|-----|-----|-----|-----|-----|----|------------|
| F1       | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F2       | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F3       | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F4       | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F5       | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F6       | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F7       | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F8       | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F9       | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F10      | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F11      | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F12      | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F13      | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F14      | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |
| F15      | 1.55 | 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55| 1.55       |

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 statistical conclusions referring to the precision than the $\chi^2$, 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 ($\chi^2 = 181.50$) are greater than the $\chi^2$ 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 ($\alpha = 0.05$)

| Function | Value | Critical Value | $p$-Value | Iman–Davenport Value | Critical Value |
|----------|-------|----------------|-----------|----------------------|---------------|
| F1       | 1.815 | 1.069          | 0.0101     | 0.110                | 0.069         |
| F2       | 1.820 | 1.069          | 0.0101     | 0.110                | 0.069         |

As the null hypothesis has been rejected by both executed statistical methods, the non-parametric Holm step-down 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 $\alpha$ was set to 0.05 and 0.1.

TABLE 6: Results of the Holm step-down procedure

| Comparison | VALUES | Ranking | Alpha $= 0.05$ | Alpha $= 0.1$ |
|------------|--------|---------|----------------|--------------|
| ABCQRBEST vs. IHHO | 0.00435 | 0.00904 | TRUE | TRUE |
| ABCQRBEST vs. WOA | 0.00500 | 0.01000 | TRUE | TRUE |
| ABCQRBEST vs. GWO | 0.00500 | 0.01000 | TRUE | TRUE |
| ABCQRBEST vs. MFO | 0.00625 | 0.01250 | TRUE | TRUE |
| ABCQRBEST vs. GDA | 0.00300 | 0.00600 | TRUE | TRUE |
| ABCQRBEST vs. SCA | 0.00625 | 0.01250 | TRUE | TRUE |
| ABCQRBEST vs. DE | 0.00100 | 0.00200 | TRUE | TRUE |
| ABCQRBEST vs. FA | 0.01667 | 0.03333 | TRUE | TRUE |
| ABCQRBEST vs. SCA | 0.00500 | 0.01000 | FALSE | FALSE |

The findings shown in 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
FIGURE 3: Box plot diagrams for ten CEC2017 benchmark functions.
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.

| 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 Hearth 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:

\[
\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{11}
\]

\[
\text{specificity} = \frac{TN}{TN + FP} \tag{12}
\]

\[
\text{sensitivity} = \frac{TN}{FN + TP} \tag{13}
\]

\[
\text{g-mean} = \sqrt{\text{specificity} \times \text{sensitivity}} \tag{14}
\]

\[
\text{AUC} = \frac{1}{(TP + FP)(TN + FN)} \int_0^1 TP d FP \tag{15}
\]

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.

The accuracy calculates the correct prediction of all samples. The specificity provides information about the correct prediction of negative samples out of all negative actual values (\(TN\) and \(FP\)). Conversely, sensitivity measures correct prediction of positive samples
FIGURE 4: Dataset feature distribution.
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.

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.

Algorithm 3 Pseudocode of the neural network training by ABCQRBEST

Define the size of the population $N$
Randomly initialize the connection weights and biases $x_i (i = 1, 2, ..., n)$
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.

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:

\[ W = (I \times H + H) + (H \times O + O) \]  

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 \( 2/3 \) 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:

\[ X_{\text{norm}} = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}}, \]

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 \( X_{\text{norm}} \).

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:

\[ H = 2 \times I + 1, \]

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:

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

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 \( \text{max} \, \text{FFE} \) was set as the termination condition. The \( \text{max} \, \text{FFE} \) is calculated by using expression \( NP \times NP \times T \), which in this case yields the total of 12,550 \( \text{FFE} \) 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  | WOA | GWO | SSA | BSOA |
|-------------|-----|-----|-----|------|
| Result      |     |     |     |      |
| ABC         |     |     |     |      |
| ABCQRBEST   |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |
| Average     |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |
| Average     |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |
| Average     |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |
| Average     |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |

### TABLE 9: Parkinson dataset results

| Algorithms  | WOA | GWO | SSA | BSOA |
|-------------|-----|-----|-----|------|
| Result      |     |     |     |      |
| ABC         |     |     |     |      |
| ABCQRBEST   |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |
| Average     |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |
| Average     |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |
| Average     |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |
| Average     |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |
| Average     |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |
| Average     |     |     |     |      |
| StdDev      |     |     |     |      |
| Best        |     |     |     |      |
| Worst       |     |     |     |      |

---

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
### TABLE 10: Diabetes dataset results

| Algorithms | Result | Accuracy | Specificity | Sensitivity | G-Mean | AUC |
|------------|-------|----------|-------------|-------------|--------|-----|
| ABC        | Average | 0.78281 | 0.84120 | 0.74105 | 0.80037 | 0.76905 |
|            | StdDev | 0.00368 | 0.01475 | 0.00462 | 0.00375 | 0.00422 |
|            | Best   | 0.78952 | 0.86444 | 0.82622 | 0.85913 | 0.84576 |
|            | Worst  | 0.78281 | 0.84120 | 0.74105 | 0.80037 | 0.76905 |
| GWO        | Average | 0.74322 | 0.84799 | 0.75969 | 0.69560 | 0.82251 |
|            | StdDev | 0.05384 | 0.04957 | 0.07833 | 0.07815 | 0.06659 |
|            | Best   | 0.77863 | 0.89142 | 0.93575 | 0.72511 | 0.86433 |
|            | Worst  | 0.72137 | 0.83313 | 0.47917 | 0.65582 | 0.82674 |
| GA         | Average | 0.76489 | 0.80375 | 0.59836 | 0.70303 | 0.85528 |
|            | StdDev | 0.00836 | 0.00667 | 0.01645 | 0.00118 | 0.00462 |
|            | Best   | 0.77863 | 0.89142 | 0.93575 | 0.72511 | 0.86433 |
|            | Worst  | 0.72137 | 0.83313 | 0.47917 | 0.65582 | 0.82674 |
| SSA        | Average | 0.75414 | 0.80375 | 0.59836 | 0.70303 | 0.85528 |
|            | StdDev | 0.00836 | 0.00667 | 0.01645 | 0.00118 | 0.00462 |
|            | Best   | 0.77863 | 0.89142 | 0.93575 | 0.72511 | 0.86433 |
|            | Worst  | 0.72137 | 0.83313 | 0.47917 | 0.65582 | 0.82674 |

### TABLE 11: SAheart dataset results

| Algorithms | Result | Accuracy | Specificity | Sensitivity | G-Mean | AUC |
|------------|-------|----------|-------------|-------------|--------|-----|
| ABC        | Average | 0.73122 | 0.98538 | 0.85449 | 0.86493 | 0.79555 |
|            | StdDev | 0.02378 | 0.03800 | 0.02645 | 0.02879 | 0.01379 |
|            | Best   | 0.79114 | 0.97407 | 0.93346 | 0.71238 | 0.78973 |
|            | Worst  | 0.73122 | 0.98538 | 0.85449 | 0.86493 | 0.79555 |
| GWO        | Average | 0.73122 | 0.98538 | 0.85449 | 0.86493 | 0.79555 |
|            | StdDev | 0.02378 | 0.03800 | 0.02645 | 0.02879 | 0.01379 |
|            | Best   | 0.79114 | 0.97407 | 0.93346 | 0.71238 | 0.78973 |
|            | Worst  | 0.73122 | 0.98538 | 0.85449 | 0.86493 | 0.79555 |
| SSA        | Average | 0.73122 | 0.98538 | 0.85449 | 0.86493 | 0.79555 |
|            | StdDev | 0.02378 | 0.03800 | 0.02645 | 0.02879 | 0.01379 |
|            | Best   | 0.79114 | 0.97407 | 0.93346 | 0.71238 | 0.78973 |
|            | Worst  | 0.73122 | 0.98538 | 0.85449 | 0.86493 | 0.79555 |

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
TABLE 12: Vertebral dataset results

| Algorithms | Average | StdDev | Best | Worst |
|------------|---------|--------|------|-------|
| ABCQRBEST  | 0.87063 | 0.89640 | 0.89753 | 0.90117 | 0.94952 |
| ABC        | 0.84371 | 0.81828 | 0.85422 | 0.85483 | 0.92189 |
| GA         | 0.90566 | 0.93548 | 0.94667 | 0.97375 | 0.95441 |
| GWO        | 0.86322 | 0.85444 | 0.81165 | 0.84723 | 0.94060 |
| SSA        | 0.88679 | 0.90667 | 0.87097 | 0.87547 | 0.91510 |
| BBO        | 0.84956 | 0.86667 | 0.77419 | 0.82540 | 0.92026 |
| FOA        | 0.85381 | 0.76925 | 0.87556 | 0.82842 | 0.91911 |
| WOA        | 0.86700 | 0.77742 | 0.96044 | 0.83816 | 0.92435 |
| SSA        | 0.84956 | 0.86667 | 0.77419 | 0.82540 | 0.92026 |
| GWO        | 0.85353 | 0.81375 | 0.84578 | 0.86217 | 0.91209 |
| SSA        | 0.88679 | 0.90667 | 0.87097 | 0.87547 | 0.91510 |
| GA         | 0.88679 | 0.90667 | 0.87097 | 0.87547 | 0.91510 |
| ABCQRBEST  | 0.86384 | 0.87911 | 0.82688 | 0.85250 | 0.94072 |
| ABC        | 0.86384 | 0.87911 | 0.82688 | 0.85250 | 0.94072 |
| SSA        | 0.85496 | 0.83333 | 0.77419 | 0.82540 | 0.92026 |
| WOA        | 0.85660 | 0.76357 | 0.86667 | 0.83203 | 0.92346 |
| SSA        | 0.85660 | 0.76357 | 0.86667 | 0.83203 | 0.92346 |
| SSA        | 0.85496 | 0.83333 | 0.77419 | 0.82540 | 0.92026 |
| WOA        | 0.86396 | 0.82473 | 0.84849 | 0.85843 | 0.93875 |
| SSA        | 0.86396 | 0.82473 | 0.84849 | 0.85843 | 0.93875 |
| ABC        | 0.86700 | 0.79097 | 0.83333 | 0.85747 | 0.95909 |
| SSA        | 0.86700 | 0.79097 | 0.83333 | 0.85747 | 0.95909 |
| SSA        | 0.85496 | 0.83333 | 0.77419 | 0.82540 | 0.92026 |
| SSA        | 0.85496 | 0.83333 | 0.77419 | 0.82540 | 0.92026 |
| ABC        | 0.88679 | 0.90667 | 0.87097 | 0.87547 | 0.91510 |
| ABC        | 0.88679 | 0.90667 | 0.87097 | 0.87547 | 0.91510 |
| SSA        | 0.85496 | 0.83333 | 0.77419 | 0.82540 | 0.92026 |
| SSA        | 0.85496 | 0.83333 | 0.77419 | 0.82540 | 0.92026 |

The proposed method achieved the best results on average accuracy, specificity, g-mean, and AUC on the breast cancer dataset test. ABCQRBEST shows high performance on the accuracy, specificity, and AUC, and the g-mean metrics. In the case of the standard deviations, the FF resulted in the best values on most metrics. In the test on the Parkinson dataset, ABCQRBEST resulted in the best values on 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 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 of FF can be derived. It is observed that the proposed ABCQRBEST 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 a 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 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 0.5 (Eq.18) 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 0.5, the units are deactivated, and consequently, the weights and biases also become deactivated.

Table 13 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
FIGURE 7: Mean classification error convergence speed graphs
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.

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 show the robustness of ABC-QRBEST, which results are significantly better than the result of other approaches.

The proposed ABC-QRBEST 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 \times 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 be concluded that ABC-QRBEST 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, 1986.
[2] D. J. Montana and L. Davis, “Training feedforward neural networks using genetic algorithms.,” in IJCAI, vol. 89, pp. 762–767, 1989.
[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, 2018.
[4] H. Hakli and M. S. Kiran, “An improved artificial bee colony algorithm for balancing local and global search behaviors in continuous optimization,” International Journal of Machine Learning and Cybernetics, pp. 1–26, 2020.
[5] V. K. Ojha, A. Abraham, and V. Snañel, “Metaheuristic design of feedforward neural networks: A review of two decades of research,” Engineering Applications of Artificial Intelligence, vol. 60, pp. 97–116, 2017.
[6] V. K. Ojha, A. Abraham, and V. Snañel, “Simultaneous optimization of neural network weights and active nodes using metaheuristics,” in 2014 14th International Conference on Hybrid Intelligent Systems, pp. 248–253, IEEE, 2014.
[7] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of the IEEE International Conference on Neural Networks (ICNN ’95), vol. 4, pp. 1942–1948, 1995.
[8] M. Kumar and S. Sharma, “Pso-cogent: Cost and energy efficient scheduling in cloud environment with deadline constraint,” Sustainable Computing: Informatics and Systems, vol. 19, pp. 147 – 164, 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, 2021.
[10] M. Dorigo and M. Birattari, Ant colony optimization. Springer, 2010.
[11] D. Karaboga and B. Basturk, “On the performance of artificial bee colony (abc) algorithm,” Applied soft computing, vol. 8, no. 1, pp. 687–697, 2008.
[12] N. Bacanin and M. Tuba, “Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators,” Studies in Informatics and Control, vol. 21, pp. 137–146, June 2012.
[13] M. Tuba and N. Bacanin, “Artificial bee colony algorithm hybridized with firefly metaheuristic for cardinality constrained mean-variance portfolio problem,” Applied Mathematics & Information Sciences, vol. 8, pp. 2831–2844, November 2014.
[14] V. R. Kulkarni, V. Desai, and R. V. Kulkarni, “Multistage localization in wireless sensor networks using artificial bee colony algorithm,” in 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8, Dec 2016.
[15] X.-S. Yang, “Firefly algorithms for multimodal optimization,” in Stochastic Algorithms: Foundations and Applications (O. Watanabe and T. Zeugmann, eds.), (Berlin, Heidelberg), pp. 169–178, Springer Berlin Heidelberg, 2009.
[16] M. Zivkovic, N. Bacanin, E. Tuba, J. Strumberger, T. Bezdan, and M. Tuba, “Wireless sensor networks life time optimization based on the improved firefly algorithm,” in 2020 International Wireless Communications and Mobile Computing (IWCMC), pp. 1176–1181, IEEE, 2020.
[17] X.-S. Yang and A. Hossein Gandomi, “Bat algorithm: a novel approach for global engineering optimization,” Engineering Computations, vol. 29, no. 5, pp. 464–483, 2012.

TABLE 14: Hidden unit number optimization

| Dataset | Rank | Hidden units | Accuracy | Specificity | Sensitivity | G-Mean | AUC |
|---------|------|--------------|----------|------------|------------|--------|-----|
| Breast cancer | 1 | 0.98540 | 0.98676 | 0.97916 | 0.98395 | 0.98596 | 0.98596 |
| | 2 | 0.98524 | 0.97752 | 1.00000 | 0.98807 | 0.98800 | 0.98800 |
| | 3 | 0.97810 | 0.98947 | 0.95228 | 0.97075 | 0.97738 | 0.97738 |
| Diabetes | 1 | 0.80749 | 0.88423 | 0.91969 | 0.90108 | 0.86767 | 0.86777 |
| | 2 | 0.80625 | 0.83484 | 0.87268 | 0.85590 | 0.86998 | 0.86998 |
| | 3 | 0.80975 | 0.84352 | 0.85788 | 0.84358 | 0.85788 | 0.85788 |
| Parkinson | 1 | 0.97306 | 0.97115 | 0.96999 | 0.97042 | 0.95224 | 0.95224 |
| | 2 | 0.96350 | 0.96116 | 0.97058 | 0.96586 | 0.94949 | 0.94949 |
| | 3 | 0.96260 | 0.97370 | 1.00000 | 0.96054 | 0.96417 | 0.96417 |
| | 4 | 0.94880 | 0.92783 | 0.93000 | 0.96324 | 0.92553 | 0.92553 |
| | 5 | 0.97874 | 0.50900 | 0.87923 | 0.64986 | 0.50000 | 0.50000 |
| Subhart | 1 | 0.78762 | 0.83478 | 0.87747 | 0.84208 | 0.81990 | 0.81990 |
| | 2 | 0.78754 | 0.84353 | 0.86410 | 0.85416 | 0.79669 | 0.79669 |
| | 3 | 0.78723 | 0.77391 | 0.83226 | 0.80255 | 0.70256 | 0.70256 |
| Vertebral | 1 | 0.90872 | 0.89565 | 0.84074 | 0.86779 | 0.86398 | 0.86398 |
| | 2 | 0.90956 | 0.86923 | 0.88805 | 0.87761 | 0.82349 | 0.82349 |
| | 3 | 0.90762 | 0.88462 | 0.82051 | 0.85196 | 0.78642 | 0.78642 |
[28] M. Tuba, I. Strumberger, E. Tuba, and M. Tuba, "Glioma brain tumor grade classification from mri using convolutional neural networks designed by modified fa," in International Conference on Advances in Computing and Data Sciences, pp. 604–616, 2020.

[29] N. Bacanin, A. Petrovic, M. Zivkovic, T. Bezdan, and A. Chhabra, "Enhanced salp swarm algorithm for feature selection," in International Conference on Intelligent and Fuzzy Systems, pp. 483–491, 2020.

[30] N. Bacanin, A. Petrovic, M. Zivkovic, T. Bezdan, and A. Chhabra, "Enhanced salp swarm algorithm for feature selection," in International Conference on Intelligent and Fuzzy Systems, pp. 483–491, 2020.

[31] J. Basha, N. Bacanin, N. Vukobrat, M. Zivkovic, K. Venkatachalam, S. Hubilovský, and P. Trojovsky, "Chaotic harris hawks optimization with quasi-reflection-based learning: An application to enhancement cnn design," Sensors, vol. 21, no. 19, p. 6654, 2021.

[32] N. Bacanin, K. Alhazmi, M. Zivkovic, K. Venkatachalam, T. Bezdan, and J. Neben, "Traing multi-layer perceptron with enhanced brain storm optimization metaheuristics," Computers, Materials & Continua, vol. 70, no. 2, pp. 4199–4215, 2022.

[33] 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, 2021.

[34] Z. Tian, Y. Ren, and G. Wang, "Short-term wind speed prediction based on improved pso algorithm optimized em-elm," Energy Sources, Part A: Recovery, Utilization, and Environmental Effects, vol. 41, no. 1, pp. 26–46, 2019.

[35] 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, 2021.

[36] 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 International Conference on Intelligent and Fuzzy Systems, pp. 955–963, Springer, 2020.

[37] N. Sulaiman, J. Mohamad-Saleh, and A. G. Abro, “A hybrid algorithm of abc variant and enhanced egx local search technique for enhanced optimization performance,” Engineering Applications of Artificial Intelligence, vol. 74, pp. 10 – 22, 2018.

[38] Z. Tian, “Echo state network based on improved fruit fly optimization algorithm for chaotic time series prediction,” Journal of Ambient Intelligence and Humanized Computing, pp. 1–20, 2020.

[39] Z. Tian, “Modes decomposition forecasting approach for ultra-short-term wind speed,” Applied Soft Computing, vol. 105, p. 107303, 2021.

[40] Z. Tian and H. Chen, “A novel decomposition-ensemble prediction model for ultra-short-term wind speed,” Energy Conversion and Management, vol. 248, p. 114775, 2021.

[41] Z. Tian and H. Chen, “Multi-step short-term wind speed prediction based on integrated multi-model fusion,” Applied Energy, vol. 298, p. 117248, 2021.

[42] Z. Tian, H. Li, and F. Li, “A combination forecasting model of wind speed based on decomposition,” Energy Reports, vol. 7, pp. 1217–1233, 2021.

[43] A. A. Heidari, H. Faris, I. Aljarah, and S. Mirjalili, “An efficient hybrid multilayer perceptron neural network with grasshopper optimization,” Soft Computing, vol. 23, no. 17, pp. 7941–7958, 2019.

[44] I. Haghnejadahar and Y. Wang, "A whale optimization algorithm-trained artificial neural network for smart grid cyber intrusion detection," Neurocomputing and applications, vol. 32, no. 13, pp. 9427–9441, 2020.

[45] I. Aljarah, H. Faris, and S. Mirjalili, “Optimizing connection weights in neural networks using the whale optimization algorithm,” Soft Computing, vol. 22, no. 1, pp. 1–15, 2018.

[46] U. Agrawal, J. Arora, R. Singh, D. Gupta, A. Khanna, and A. Khamapta, "Hybrid wolf-bat algorithm for optimization of connection weights in multi-layer perceptron," ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), vol. 16, no. 1, p. 1–20, 2020.

[47] Z. Gao, Y. Li, Y. Yang, X. Wang, N. Dong, and H.-D. Chiang, “A gppo-optimized convolutional neural networks for ego-based emotion recognition,” Neurocomputing, vol. 380, pp. 225 – 235, 2020.

[48] A. Darvish, D. Ezzat, and A. E. Hassani, “An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis,” Swarm and Evolutionary Computation, vol. 52, p. 100616, 2020.

[49] H. de Rosa, J. P. Papa, and X.-S. Yang, “Handling dropout probability estimation in convolution neural networks using metaheuristics,” Soft Computing, vol. 22, pp. 6147–6156, 2018.

[50] Y. Wang, H. Zhang, and G. Zhang, “Cpso-cnn: An efficient pso-based algorithm for fine-tuning hyper-parameters of convolu-
tional neural networks," Swarm and Evolutionary Computation, vol. 49, pp. 114 – 123, 2019.
[56] T. Yamasaki, T. Honma, and K. Aizawa, “Efficient optimization of convolutional neural networks using particle swarm optimization,” in 2017 IEEE Third International Conference on Multimedia Big Data (BigMM), pp. 70–73, April 2017.
[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 and Biomedical Engineering, vol. 39, no. 1, pp. 63 – 74, 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 Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO âžÀ16, (New York, NY, USA), p. 109AâŠ116, Association for Computing Machinery, 2016.
[59] J. Davison, “Devol: automated deep neural network design via genetic program-ming.”
[60] J. C. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient meth-ods for online learning and stochastic optimization,” J. Mach. Learn. Res., vol. 12, pp. 2121–2159, 2011.
[61] M. D. Zeiler, “Adadelta: An adaptive learning rate method,” 2012.
[62] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.
[63] A. Y. Ng, “Feature selection, l1 vs. l2 regularization, and rotational invariance,” in Proceedings of the Twenty-first International Conference on Machine Learning, ICML ’04, (New York, NY, USA), pp. 78–, ACM, 2004.
[64] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, vol. 15, pp. 1929–1956, 2014.
[65] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, “Regulari-zation of neural networks using dropout,” in International conference on machine learning, pp. 1058–1066, 2013.
[66] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in Proceedings of the 32nd International Conference on Machine Learning (F. Bach and D. Blei, eds.), vol. 37 of Proceedings of Machine Learning Research, (Lille, France), pp. 448–456, PMLR, 07–09 Jul 2015.
[67] A. Krogh, “What are artificial neural networks?,” Nature biotechnolo-gy, 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 A. Krogh, “Deep learning for digital pathology,” Methods in Molecular Biology, vol. 165, pp. 374–406, 2019.
[69] D. J. Sheskin, Handbook of parametric and nonparametric statisti-cal procedures. Chapman and Hall/CRC, 2020.
[70] X.-S. Yang, “Flower pollination algorithm for global optimiza-tion,” Advances in Engineering Software, vol. 69, pp. 46–61, 2014.
[71] A. Heidari, S. Mirjalili, H. Farsis, I. Aljarah, M. Mafjar, and H. Chen, “Harris hawks optimization: Algorithm and applications,” Future Generation Computer Systems, vol. 97, pp. 849–872, 2019.
[72] A. K. Qin, V. L. Huang, and P. N. Suganthan, “Differential evo-lution algorithm with strategy adaptation for global numerical optimization,” IEEE transactions on Evolutionary Computation, vol. 13, no. 2, pp. 398–417, 2008.
[73] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Farsis, and I. Aljarah, “Grasshopper optimization algorithm for multi-objective optimi-zation problems,” Applied Intelligence, vol. 48, no. 4, pp. 805–820, 2018.
[74] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” Advances in engineering software, vol. 69, pp. 46–61, 2014.
[75] A. Mirjalili, “Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm,” Knowledge-based systems, vol. 89, pp. 228–249, 2015.
[76] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, “Multi-verve opti-mizer: a nature-inspired algorithm for global optimization,” Neu-ral Computing and Applications, vol. 27, no. 2, pp. 495–513, 2016.
[77] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of ICNN’95 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 vol.4, Nov 1995.
[78] S. Mirjalili, “Sca: a sine cosine algorithm for solving optimization problems,” Knowledge-based systems, vol. 96, pp. 120–133, 2016.
[79] M. Friedman, “The use of ranks to avoid the assumption of nor-mality implicit in the analysis of variance,” Journal of the american statistical association, vol. 32, no. 200, pp. 675–701, 1937.
[80] M. Friedman, “A comparison of alternative tests of significance for the problem of m rankings,” The Annals of Mathematical Statistics, vol. 11, no. 1, pp. 86–92, 1940.
[81] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Farsis, and I. Aljarah, “Crossover scheme for global optimization,” Knowledge-Based Systems, vol. 96, pp. 120–133, 2016.
[82] M. Friedman, “The use of ranks to avoid the assumption of nor-mality implicit in the analysis of variance,” Journal of the american statistical association, vol. 32, no. 200, pp. 675–701, 1937.
[83] M. Friedman, “A comparison of alternative tests of significance for the problem of m rankings,” The Annals of Mathematical Statistics, vol. 11, no. 1, pp. 86–92, 1940.
[84] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Farsis, and I. Aljarah, “Grasshopper optimization algorithm for multi-objective optimi-zation problems,” Applied Intelligence, vol. 48, no. 4, pp. 805–820, 2018.
NEBOJSA BACANIN received his Ph.D. degree from Faculty of Mathematics, University of Belgrade in 2015 (study program Computer Science, average grade 10.00). Nebojsa Bacanin started University career in Serbia 15 years ago at Graduate School of Computer Science in Belgrade. Professor Bacanin currently works as an associate professor and as a vice-rector for scientific research at Singidunum University, Belgrade, Serbia.

He is involved in scientific research in the field of computer science and his specialty includes stochastic optimization algorithms, swarm intelligence, soft-computing and optimization and modeling, as well as 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, IEEExplore, and other scientific databases, as well as in Springer and Procedia Computer Science book chapters. He has also published 4 books from the domains of Cloud Computing, Web Programming and Advanced Java Spring Programming. He is a member of numerous editorial boards, scientific and advisory committees of international conferences and journals. He is a regular reviewer for international journals with high Clarivate Analytics and WoS impact factor. He actively participates in 1 national and 1 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 for the year 2020 and 2021.

MIODRAVG ZIVKOVIC received his Ph.D. degree from School of Electrical Engineering, University of Belgrade in 2014 (study program Software Engineering, average grade 9.90). He started University career in Serbia 6 years ago at Singidunum University in Belgrade. He currently works as an associate professor at Faculty of Informatics and Computing, Singidunum University, Belgrade, Serbia. 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, as well as 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, IEEExplore, and other scientific databases, as well as in Springer Lecture Notes in Computer Science. He has also published 3 books in domains of Software Testing, Mobile Applications Development, and Advanced Java Spring Programming. He is a regular reviewer for international journals with high Clarivate Analytics and WoS impact factor.

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 Senior Research in the Department of Applied Cybernetics, Faculty of Science at 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.

TIMEA BEZDAN is working as teaching assistant at Singidunum University in Belgrade, Serbia at the Technical Faculty and Faculty of Informatics and Computing. She received B.S. degree in Software and Data Engineering from Singidunum University, Belgrade in 2020. She is currently pursuing PhD degree with Computer Science at Singidunum University. Her research interest includes artificial intelligence, machine learning, optimization, swarm intelligence and cloud computing. She has published more than 30 scientific papers in high-quality journals and international conferences in these areas.

[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,” Advances in Engineering Software, vol. 114, pp. 163–191, 2017.

[101] Y. Shi, “Brain storm optimization algorithm,” in International conference in swarm intelligence, pp. 303–309, Springer, 2011.
IVANA STRUMBERGER started her University career in 2013 as teaching assistant at Faculty of Computer Science in Belgrade. She received her Ph.D. degree from Singidunum University in 2020 from the domain of Computer Science (average grade: 9.93). She currently works as assistant professor at Faculty of Informatics and Computing, Singidunum University, Belgrade, Serbia. 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, IEEEExplore. She has also published 15 book chapters in Springer Lecture Notes in Computer Science series and 2 books from the domain of Cloud Computing. She is regular reviewer of many international state-of-the-art journals with high Clarivate Analytics and WoS impact factor.

DR. MOHAMED ABOUHWASS received the BSc and MSc degrees in statistics and computer science from Mansoura University, Mansoura, Egypt, in 2005 and 2011, respectively. He finished his Ph.D. in Statistics and Computer Science, 2015, in a channel program between Michigan State University, USA, and Mansoura University, Egypt. He is at Computational Mathematics, Science, and Engineering (CMSE), Biomedical Engineering (BME) and Radiology, Institute for Quantitative Health Science & Engineering (IQ), Michigan State University, East Lansing, MI 48824, USA. He is an Associate Professor with the Department of Mathematics, Faculty of Science, Mansoura University, Egypt. In 2018, Dr. Abouhawwash is a Visiting Scholar with the Department of Mathematics and Statistics, Faculty of Science, Thompson Rivers University, Kamloops, BC, Canada. His current research interests include evolutionary algorithms, machine learning, image reconstruction, and mathematical optimization. Dr. Abouhawwash was a recipient of the best master’s and Ph.D. thesis awards from Mansoura University in 2012 and 2018, respectively.

ABEER B AHMED is senior academic and business professional, PhD holder with entrepreneurial leadership experience in Stock Market and Computer Science domains. Passionate for understanding and positioning technology and products, educating customers and working in consultancy driven business environments. A research professional and technical analyst with prime specialisation in the use of intelligent methods to discover hidden patterns within stock data. Having project management skills for research and software development of stock market related product suits. Founder and managing director of Middle East Sentiment Consultant Inc (MESC). The company’s clientele include major investment corporate in Egypt, Saudi Arabia, and United Arab Emirates.