***The proposed metaheuristic optimization algorithm based on the two-step Adams-Bashforth scheme (MOABT) was used in this paper for Multilayer Perceptron Training (MLP). In computer science and mathematical examples, metaheuristic is high-level procedures or guidelines designed to find, devise, or select algorithmic research methods to obtain high-quality solutions to an example problem, especially if the information is insufficient or incomplete, or if computational capacity is limited. Many metaheuristic methods include some stochastic example operations, which means that the resulting solution is dependent on the random variables that are generated during the search. The use of higher evidence can frequently find good solutions with less computational effort than iterative methods and algorithms because it searches a broad range of feasible solutions at the same time. Therefore, metaheuristic is a useful approach to solving example problems. There are several characteristics that distinguish metaheuristic strategies for the research process. The goal is to efficiently explore the search perimeter to find the best and closest solution. The techniques that make up metaheuristic algorithms range from simple searches to complex learning processes. Eight model data sets are used to calculate the proposed approach, and there are five classification data sets and three proximate job data sets included in this set. The numerical results were compared with those of the well-known evolutionary trainer Gray Wolf Optimizer (GWO). The statistical study revealed that the MOABT algorithm can outperform other algorithms in terms of avoiding local optimum and speed of convergence to global optimum. The results also show that the proposed problems can be classified and approximated with high accuracy***

***Keywords: algorithm, Adams-Bashforth method, approximation, classification, global, metaheuristic, multilayer, perceptron, training, optimization***

UDC 519
DOI: 10.15587/1729-4061.2022.254023

# METAHEURISTIC OPTIMIZATION ALGORITHM BASED ON THE TWO-STEP ADAMS-BASHFORTH METHOD IN TRAINING MULTI-LAYER PERCEPTRONS

**Hisham M. Khudhur**
Lecturer
Department of Mathematics
College of Computer Science and Mathematics
University of Mosul
University str., 43, Ninawa, Mosul, Iraq, 41002
**Kais I. Ibraheem**
*Corresponding author*
Assistant Professor, Dean of College
Department of Computer Science
College of Education for Pure Science
University of Mosul
Alkendy str., 34, Ninawa, Mosul, Iraq, 41002
E-mail: kaisismail@uomosul.edu.iq

*Received date 20.01.2022*
*Accepted date 23.02.2022*
*Published date 28.04.2022*

***How to Cite:*** *Khudhur, H. M., Ibraheem, K. I. (2022). Metaheuristic optimization algorithm based on the two-step Adams-Bashforth method in training multi-layer perceptrons. Eastern-European Journal of Enterprise Technologies, 2 (4 (116)), 6–13. doi: https://doi.org/10.15587/1729-4061.2022.254023*

## 1. Introduction

Artificial Neural Networks (ANNs) are the most actively explored area in the fields of Artificial Intelligence and Machine Learning (ANN). In numerous domains, including pattern recognition, regression, classification, signal processing, robotics, image, audio, and signal identification, optimization, and data clustering, artificial neural networks (ANNs) have been successfully implemented [1–4].

In essence, neural networks are parallel computing devices, and they are being used in an effort to create a computer model of the brain. The primary goal is to design a system that can do a variety of computing tasks more quickly than traditional systems. The importance of graphic neural networks is that a lot of real-world data can be represented in graphic form, such as social networks, chemical compounds, maps, transportation systems, and others. Nodes in these networks exchange information with neighboring nodes, enabling them to learn. The training of artificial neural networks is required in order to achieve good output values for the data that we have entered into them [5]. The purpose of the training process is to identify the ideal weights and biases for the neural network based on the data that is fed into it. Training neural networks has the goal of decreasing the error between the output of the network and the target. The weights and biases of the network are altered during the training process. The training procedure for artificial neural networks has a direct impact on the performance of the networks.

Therefore, studies that are devoted to the classification of different data, as well as to the approximation of mathematical functions, are of scientific relevance. These studies are related to the development of training algorithms for artificial neural networks.

## 2. Literature review and problem statement

In previous researches, many deterministic methods were presented to train neural networks. Avoid stochastic deterministic methods to produce the same output for the same input. Gradient-based methods make up most of the

deterministic methods. Derivation of the objective function from which gradient-based methods make use. When the objective function has an optimal local value, these methods globally do not guarantee an optimal solution to the problems. The backpropagation algorithm and its variations are well known and are an example of gradient-based techniques [6]. Gradient-based technologies are speed and simple. However, the disadvantages are their staying tendency at the local optimum level [7], initial parameters dependency, and their convergence is at early and slow stages [8]. Metaheuristic algorithms were proposed as an alternative to gradient-based methods for training artificial neural networks. By randomness, metaheuristic algorithms start in the training phase and reduce the error over time. Metaheuristic algorithms are better at global optimization [9]. These strategies are particularly effective in avoiding local optimizations. Despite this, they are often more time-consuming than deterministic methods [10]. When the problem got more sophisticated and multidimensional, metaheuristic approaches were included in the ANN training process, and it was discovered that these algorithms were superior to the gradient-based methodology in terms of accuracy [11]. Metaheuristic optimization strategies based on swarm intelligence are a critical component of metaheuristic methodologies [12]. Simplicity, parallelism, and application to a wide range of optimization problems such as real parameter optimization (RPO), combinatorial optimization, and mixed integer optimization (MIMO) are some of the characteristics of metaheuristics, and they have been shown to be effective in a wide range of real-world and engineering problems [13]. In order to provide the most up-to-date empirical research on metaheuristic methodology and approaches for future system advancements, Advancements in Applied Metaheuristic Computing is the journal to consult. It also presents outcomes obtained via the use of optimization methods. Intelligent algorithms are used to simulate animal species that act intelligently in groups, such as birds and other animals, in order to create realistic simulations [14] introduces structural optimization problems. A benchmark nonlinear constrained optimization problem is used to test the new CS method with Lévy flights, swarms of wolves [15]. Derived from the natural leadership structure and hunting behavior of grey wolves, swarms of whales [16]. Whale Optimization Algorithm (WOA), which simulates humpback whale social behavior. The algorithm is based on bubble-net hunting, swarms of fireflies [17]. This study describes a novel Firefly Algorithm (FA) for multimodal optimization, and bat swarms [18] propose the bat algorithm (BA), a novel nature-inspired metaheuristic optimization technique for engineering optimization. Every swarm agent (individual) makes a possible solution suggestion by sharing information with other agents in the swarm, each self-organized agent will attempt to discover the optimum solution [9]. Particle Swarm Optimization (PSO) is one of several intelligent methods for ANN training that have been reported in the literature. PSO is an intelligent technique that uses optimization of three-layer feed-forward artificial neural network (ANN) structure and parameters (weights and bias) [19]. Cuckoo swarms Technique is well-suited for the solution of optimization issues [20], bat swarms Optimization Technique [21]. The bat algorithm's benefit is its population-based algorithm and local search. Grey Wolf swarms Optimization Technique [22] originally suggested Grey Wolf Optimizer (GWO) for MLP training, Whale swarms Optimization Technique [7]. The nonlinear structure of neural networks makes training them challenging (weights and biases). Firefly swarms Technique [23]. It is discovered that the suggested strategy generates a more consistent convergence, Grasshopper swarms Optimization Technique [24]. This research proposes a novel hybrid stochastic training approach for multilayer perceptrons (MLPs) neural networks, and Dragonfly swarms Technique [25]. Social interactions in DA may lead to poor solution accuracy, easy local optima stalling, and an imbalance between exploration and exploitation. Although several ANN training methods have been published in the literature, new techniques are required to overcome issues such as the local minimum reach and early convergence, which are currently being debated. Any optimization strategy for all optimization problems cannot be solved, according to the No-Free-Lunch Theorem (NFL) [26]. In this paper, we use mathematical methods in metaheuristic instead of the well-known swarms such as the swarms of ants, gray wolves, cuckoos, black monkeys, etc.

## 3. The aim and objectives of the study

The aim of the study is to suggest metaheuristic techniques for training artificial neural networks based on the two-step Adam Bashforth's method.

To achieve the aim, the following objectives were set:

– to get the least mean square error;

– to classify data accurately and efficiently;

– to approximate the functions more precisely.

## 4. Materials and methods of research

### 4. 1. Two-Step Adams-Bashforth Method

The derivation for the Adams-Bashforth family of numerical methods is well-known, but I couldn't find a source that supplied the two-step approach where the two step sizes are different. Two beginning points are needed in the two-step approach. An Euler step is frequently used to determine the second point, however, because the Euler approach is inefficient, it is rarely used $O(h^1)$. To prevent creating a massive global error, I want to keep this first step simple. The Adams-Bashforth method is then used to compute the third point using varied step sizes. After then, you can utilize the Adams-Bashforth approach as usual. Another application could be in an adaptive step size approach, in which the step sizes are adjusted as the process progresses.

Euler's approach is a simple one-step procedure. The two-step Adams-Bashforth method is a basic multistep procedure:

$$y_{n+2} = y_{n+1} + \frac{3}{2}hf\left(t_{n+1}, y_{n+1}\right) - \frac{1}{2}hf\left(t_n, y_n\right). \quad (1)$$

To compute the following value, $y_{n+2}$, this approach requires two values, $y_{n+1}$ and $y_n$. The starting value problem, on the other hand, only supplies one value, $y_0=1$. Using $y_1$ computed by the Euler's method as the second value is one way to tackle this problem. The Adams-Bashforth method produces the following results with this decision [27, 28].

### 4. 2. Feed-forward neural network and multi-layer perceptron (MLP)

Neural network systems that feed information from input to output are referred to as feed-forward neural network

systems (NN). A feed-forward neural network (NN) is a computational information network that operates in a single direction. It has an I-H-O design, where I and O represent the input and output layers, respectively, and H represents the hidden layer. MLP is the most widely used type of feed-forward neural network (s).

The fitness function is what determines the neuronal error. The following is the process for calculating fitness functions.

The first two layers of MLP represent biases and weights. As a result, (2) gives total neural input:

$$s_p = \sum_{i=1}^{n} w_{ip} \cdot x_i - \theta_p, \ p = 1,2,\ldots,n, \tag{2}$$

where $x$ is the input and $w$ is the MLP's weight. Furthermore, $\theta_p$ represents prejudice.

The sigmoid function determines the fitness of inputs in MLP, as indicated in (3).

$$S_p = \text{sigmoid}\left(s_p\right) = \frac{1}{1+\exp\left(-s_p\right)}, \ p = 1,2,\ldots,n. \tag{3}$$

The output of the trained MLP is now calculated using (4), (5).

$$O_l = \sum_{i=1}^{h} w_{pl} \cdot s_p - \theta_l, \ l = 1,2,\ldots,m, \tag{4}$$

$$O_l = \frac{1}{1+\exp\left(-o_l\right)}, \ l = 1,2,\ldots,m. \tag{5}$$

The weights are for determining the final output of MLPs for given inputs, as shown in (2) to (5). The technical definition of MLP training is the identification of appropriate biases and weights to produce the desired relationship between inputs and outputs. The MOABT algorithm is used as an MLP trainer in the following sections.

### 4. 3. Brief description of the MOABT algorithm

#### 4. 3. 1. Initialization stage

The logic of this stage is to create an initial swarm that evolves over a specified number of iterations until the stage is complete. In MOABT, for a population of size $N$, a total of $N$ sites are generated at random. $x_n=(n=1, 2,..., N)$, where $n=1, 2,..., N$, denotes the number of individuals in the population who are $D$ dimensional optimization problem solutions. On a general level, the initial positions are generated at random using the concept described below:

$$x_{n,l} = L_l + \text{rand} \cdot \left(U_l - L_l\right).$$

It is the $l^{\text{th}}$ variable in the problem ($l=1, 2,..., D$), which has lower and upper limits of $L_l$ and $U_l$, respectively, and is a random number in the range of [0, 1]. This rule generates only a small number of solutions, compared to other rules.

#### 4. 3. 2. Search Mechanism in MOABT Phase

Adams-Bashforth is a two-step process. The method employed in this study is used to search the decision space and construct an appropriate global and local search strategy. This method, which is based on the Adams-Bashforth two-step procedure, was used to determine the proposed search mechanism for MOABT [29].

The following is the definition of the SM formula:

$$k_1 = \frac{1}{2\Delta X}\left(rand \times X_w - u \times X_b\right),$$

$$k_2 = \frac{1}{2\Delta X}\begin{pmatrix} rand \times \left(x_w + rand_1 \times k_1 \times \Delta X\right) - \\ -\left(uOX_b + rand_2 \times k_1 \times \Delta X\right) \end{pmatrix},$$

$$u = round\left(1+rand\right) \times \left(1-rand\right),$$

$$X_{RK} = -0.5k_1 + 1.5k_2,$$

$$SM = \Delta X\left(X_{RK}\right).$$

A random value ranging from zero to one is represented by $rand_1$ and $rand_2$. The value of $\Delta x$ is defined by the following formula [29]:

$$\Delta X = 2 \times rand \times \left|Stp\right|,$$

$$Stp = rand \times \left(\left(X_b - rand \times X_{avg}\right) + \gamma\right),$$

$$\gamma = rand \times \left(X_n - rand \times \left(u - l\right)\right) \times \exp\left(-4 \times \frac{i}{\max_i}\right).$$

In this study, $\mathbf{X}_w$ and $\mathbf{X}_b$ are determined as follows:

if $w\left(X_n\right) < w\left(X_{bi}\right)$.

$X_b = X_n$;

$X_w = X_{bi}$;

else

$X_b = X_{bi}$;

$X_w = X_n$;

end.

#### 4. 3. 3. Updating solutions

Using a search mechanism, the MOABT technique, which is based on the two-step Adams-Bashforth approach, shifts the current solution position on each iteration, resulting in a more accurate solution position (SM):

If $rand<0.5$;
**(stage of discovery)**

$$X_{n+1} = \left(X_c + r \times SF \times g \times X_c\right) + \\ + SF \times SM + \mu \times randn \times \left(X_m - X_c\right);$$

else
**(*exploitation phase*)**

$$X_{n+1} = \left(X_m + r \times SF \times g \times X_m\right) + \\ + SF \times SM + \mu \times randn \times \left(X_{r1} - X_{r2}\right); \tag{6}$$

end.

The integer $r$ can be either 1 or 1, depending on the situation. $g$ is a random number in the range of 0 to 2. $SF$ is a coping mechanism for many people. $\mu$ is a random number generated by the computer.

*SF* is calculated using the following formula:

$$SF = 2\times(0.5 - rand)\times u, \tag{7}$$

$$f = a\times exp\left(-b\times rand\times\left(\frac{i}{\max_i}\right)\right), \tag{8}$$

where $\max_i$ denotes the number of iterations that have been performed. These are the $\mathbf{X}_c$ and $\mathbf{X}_m$ expressions in formula form:

$$\mathbf{X}_c = \varphi\times\mathbf{X}_n + (1-\varphi)\times\mathbf{X}_{r1}; \tag{9}$$

$$\mathbf{X}_m = \varphi\times\mathbf{X}_{best} + (1-\varphi)\times\mathbf{X}_{lbest}. \tag{10}$$

Here, the random number between 0 and 1 is denoted by the letter φ. As of now, $X_{best}$ has proven to be the most effective solution available. $X_{best}$ represents the best position at the end of each iteration [29].

**4. 3. 4. Enhanced solution quality (ESQ)**

Each iteration of the MOABT algorithm employs the Enhanced Solution Quality (ESQ) technique, which is designed to improve the quality of the solutions while simultaneously reducing local optimization (ESQ). Construction of the answer ($X_{new2}$) with the ESQ is accomplished via the usage of the following approach:

if $rand < 0.5$,
if $u < 1$,

$$\mathbf{X}_{new2} = \mathbf{X}_{new1} + r\times u\times\left|\left(\mathbf{X}_{new1} - \mathbf{X}_{avg}\right) + randn\right|,$$

else

$$\mathbf{X}_{new2} = \left(\mathbf{X}_{new1} - \mathbf{X}_{avg}\right) + r\times u\times\left|\left(u\cdot\mathbf{X}_{new1} - \mathbf{X}_{avg}\right) + randn\right|, \tag{11}$$

end.
end.

$$u = rand(0.2)\times e^{\left(-c\left(\frac{i}{\max_i}\right)\right)}, \tag{12}$$

$$\mathbf{X}_{avg} = \frac{\mathbf{X}_{r1} + \mathbf{X}_{r2} + \mathbf{X}_{r3}}{3}, \tag{13}$$

$$\mathbf{X}_{new1} = \beta\times x_{avg} + (1-\beta)\times\mathbf{X}_{best}. \tag{14}$$

where β is a random number between 0 and 1, and where for the sake of this paper, the random number *c* is equal to 5 rand, $X_{best}$ is the best solution that has been discovered so far, *r* is an integer that may be one of the following values: 1, 0, or –1.

It is possible that the current answer (i.e., $w(X_{new2}) > w(X_n)$) is not as good as the solution found in this section in terms of fitness ($X_{new2}$). It is decided to construct another new solution ($X_{new2}$) in order to have a second chance at generating a workable solution. The following is an explanation of what it is:

if $rand < u$;

$$\mathbf{X}_{new3} = \left(\mathbf{X}_{new2} - rand\times\mathbf{X}_{new2}\right) + SF\times\left(rand\times\mathbf{X}_{RK} + \left(v\times\mathbf{X}_b - \mathbf{X}_{new2}\right)\right), \tag{15}$$

end,

where *v* is a two-digit random number multiplied by the number of *rand* in the game [29].

Algorithm 1. The MOABT pseudo-code.
Phase One. Initialization.
Set the variables *a* and *b* to their default values before continuing.
It is necessary to construct the MOABT population $X_i = (i = 1, 2, ..., I)$.
Determine the objective function of each member of the population.
Find the $X_w$, $X_b$, and $X_{best}$ solutions to your problems.
Phase Two. MOABT's operational toe.
for $k = 1$:max $k$;
for $i = 1:I$;
for $j = 1:J$.
To find the location of $X_{i+1,j}$, the equation (1) is employed.
end for
ESQ
if $rand < 0.5$.
Eq. (11) may be used to determine the location of $X_{new2}$
if $w(X_i) < w(X_{new2})$;
if $rand < u$.
Eq. (15) may be used to determine the location of $X_{new3}$
end
end
end.
Positions $X_u$ and $X_b$ should be adjusted as a result of this.
$k = k+1$
end
Phase Three. $X_{best}$ should be returned.

The above algorithm summarizes the work of the new technique used to train artificial neural networks to classify data and approximate functions.

**4. 4. MOABT-based MLP trainer**

The purpose of the model proposed is to determine the appropriate biases and weights for MLP training so that a small test error can be achieved, and a high rating as well. In this model, an MLP learner is achieved by using the MOABT algorithm.

The problem representation is the first and most crucial stage in training a metaheuristic MLP [30]. To put it another way, the training of MLPs' problem must be phrased in such a way that it can be solved via meta-inference. Biases and weights are the most crucial components of MLP training, as stated in the beginning. The coach must figure out which biases and weights produce the most accurate classification, approximation, and prediction.

As a result, the weights and biases are the variables. The variables of an MLP are supplied in the following format for this method because the MOABT algorithm sets the variables in a vector:

$$\vec{V} = \left\{\vec{W}, \vec{\theta}\right\} = \left\{W_{1,1}, W_{1,2}, \cdots, W_{n,n}, h, \theta_1, \theta_2, \cdots, \theta_h\right\}, \tag{16}$$

where *n* is the number of input nodes, the connection weight from the $i^{th}$ node to the $j^{th}$ node is represented by $W_{ij}$, and $\theta_j$ is the $j^{th}$ hidden node's bias (threshold).

The objective function of the MOABT algorithm must be determined after determining the variables. MLP training aims to achieve the maximum possible approximation, classification, or prediction accuracy for both training and test samples, as mentioned earlier. In order to evaluate MLP, the mean squared error is the standard measurement (MSE). The MLP training sequence is employed in this measurement,

and the difference between the target and the output value provided by the MLP is calculated using the formula below:

$$MSE = \sum_{i=1}^{m} \left(o_i^k - d_i^k\right)^2, \quad (17)$$

where $o_i^k$ is the exact output of the $i^{\text{th}}$ input unit when the $k^{\text{th}}$ training sample occurs in the input, $d_i^k$ is the preferred output of the $i^{\text{th}}$ input unit when the $k^{\text{th}}$ training sample is utilized and $m$ is the number of outputs.

The MLP must clearly be tailored to a wide range of training data for efficiency. As a result, MLP performance is assessed using the average MSE across all training samples:

$$\overline{MSE} = \sum_{k=1}^{s} \frac{\sum_{i=1}^{m} \left(o_i^k - d_i^k\right)^2}{s}, \quad (18)$$

where $s$ denotes the training samples, $m$ denotes the number of outputs, $d_i^k$ denotes the required output of the $i^{\text{th}}$ input unit as the $k^{\text{th}}$ sample is used for training, and $o_i^k$ denotes the exact output of the $i^{\text{th}}$ input unit as the $k^{\text{th}}$ training sample of the input.

Following all the MOABT algorithm's variables and average MSE, which can be used to phrase the problem of training an MLP as follows:

$$\text{Minimize}: F\left(\vec{V}\right) = \overline{MSE}. \quad (19)$$

By repeatedly modifying the MLP biases and weights to minimize the mean MSE, MOABT can be converged to a global solution better than random starting solutions. As a result, each iteration alters weights and biases, as well as shifting locations.

In order to train MLP and estimate the average MSE, the MOABT approach looks for the optimum biases and weights to employ with training samples. This procedure, as indicated in Fig. 1, continues in iterations until the best solution (i.e. minimum MSE) is identified.

In Fig. 1 we show the linkage of the method with the advanced method with artificial neural networks.

For verification, the findings are compared to GWO [22]. The optimization process is supposed to start with the production of random weights and biases in the range of [10, 10] for all data sets. There are eight training/test samples, three characteristics, and two classes in the XOR data set, which is shown in [22].

The balloon dataset comprises four characteristics, 16 training samples, 16 test samples, and two classes, making it more complex than XOR. Only four characteristics and 150 training/test samples are included in the Iris dataset. The breast cancer dataset contains 599 training samples, 9 characteristics, 100 test samples, and 2 classes of breast cancer disease. All 22 features, 187 test samples and 80 training samples are part of the heart dataset. These rating datasets were specifically chosen to give a variety of training/test samples and degrees of complexity in order to adequately measure the efficacy of a MOABT-based MLP trainer. The sigmoid is the simplest to approximate, while the sine on the contrary is the most complex, for more see [22].

Matlab 2021b software and an HP laptop with a hard drive of 512 GB and 8 GB RAM, as well as a Windows 10 operating system, were used to develop, simulate, and train all of the problems.

## 5. Results of neural network training

### 5. 1. Mean Square Error (MSE)

Using the MOABT algorithm, the mean square error of all the given problems has been reduced and compared with the GWO algorithm as can be seen in Table 1 and Fig. 2–9.

Table 1

Problems experimental results

| MOABT | GWO | iterations | problem |
|---|---|---|---|
| 3.1889e-09 | 5.7052e-05 | 500 | XOR |
| 2.8246e-30 | 1.3428e-13 | 500 | Balloon |
| 0.66667 | 0.66669 | 500 | Iris |
| 0.00094204 | 0.0014795 | 500 | Breast cancer |
| 0.053138 | 0.07194 | 500 | Heart |
| 0.24632 | 0.24635 | 500 | Sigmoid |
| 0.17533 | 0.17614 | 500 | Cosine |
| 0.40018 | 0.45226 | 500 | Sine |

The training results are shown in Table 1. Here we conclude the section on the numerical results of training artificial neural networks.

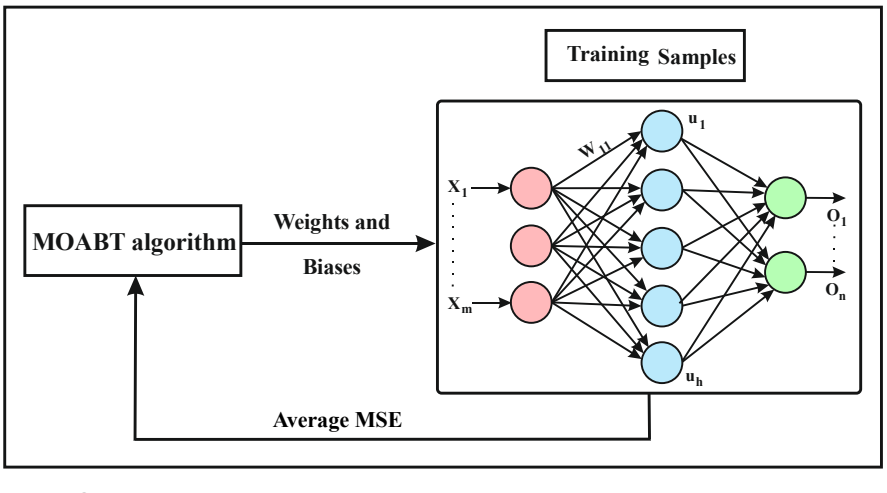


Fig. 1. The MOABT algorithm looks for weights and biases in the training samples in order to train the Multilayer Perceptron (MLP) and determine the average mean square error (MSE)

### 5. 2. Data Classification

We obtained high efficiency and accuracy in classifying the data of XOR, Balloon, Iris, Breast Cancer, and Heart in comparison with the GWO algorithm, where it clearly outperformed it as shown in Table 1 and Fig. 2–6.

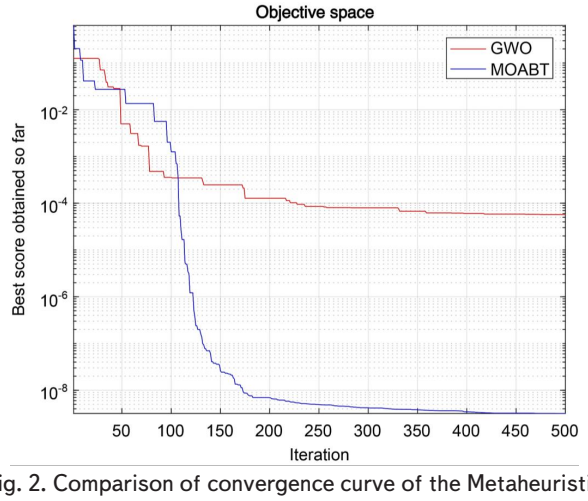


Fig. 2. Comparison of convergence curve of the Metaheuristic optimization algorithm based on the Adams-Bashforth two-step scheme (MOABT) technique and GWO in problem 1

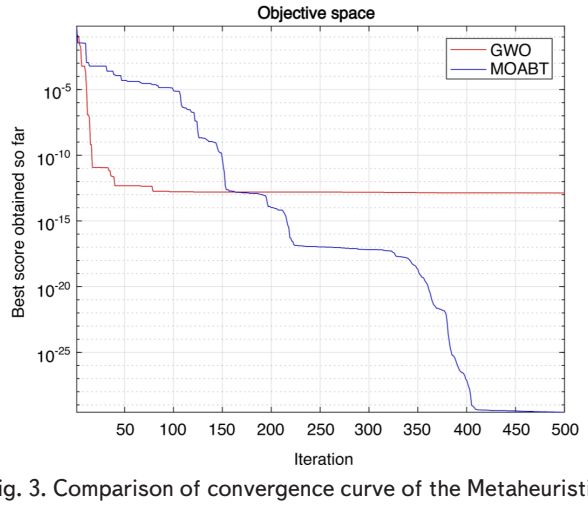


Fig. 3. Comparison of convergence curve of the Metaheuristic optimization algorithm based on the Adams-Bashforth two-step scheme (MOABT) technique and GWO in problem 2

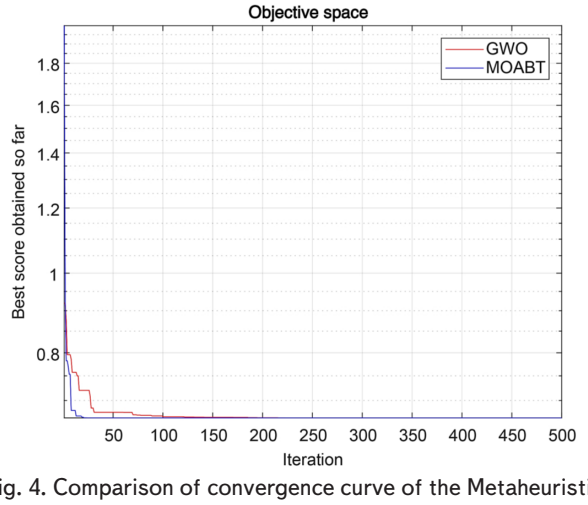


Fig. 4. Comparison of convergence curve of the Metaheuristic optimization algorithm based on the Adams-Bashforth two-step scheme (MOABT) technique and GWO in problem 3

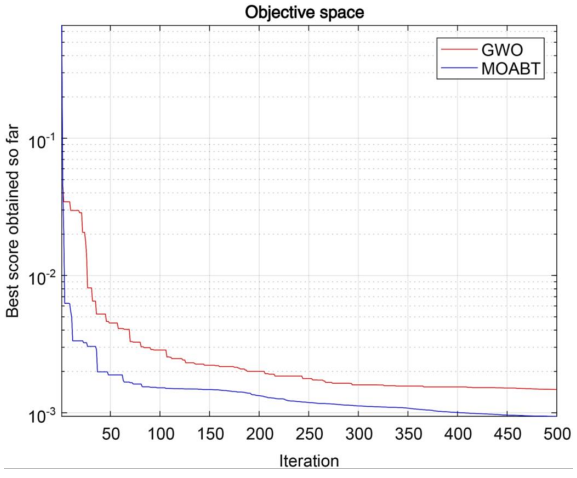


Fig. 5. Comparison of convergence curve of the Metaheuristic optimization algorithm based on the Adams-Bashforth two-step scheme (MOABT) technique and GWO in problem 4

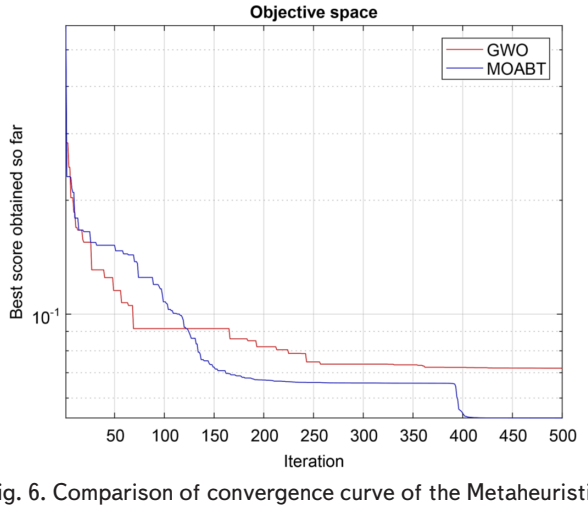


Fig. 6. Comparison of convergence curve of the Metaheuristic optimization algorithm based on the Adams-Bashforth two-step scheme (MOABT) technique and GWO in problem 5

### 5. 3. Approximation Functions

We obtained high efficiency and accuracy in approximating Sigmoid, Cosine, and Sine functions compared to the GWO algorithm, it clearly outperformed it as shown in Table 1 and Fig. 7–9.

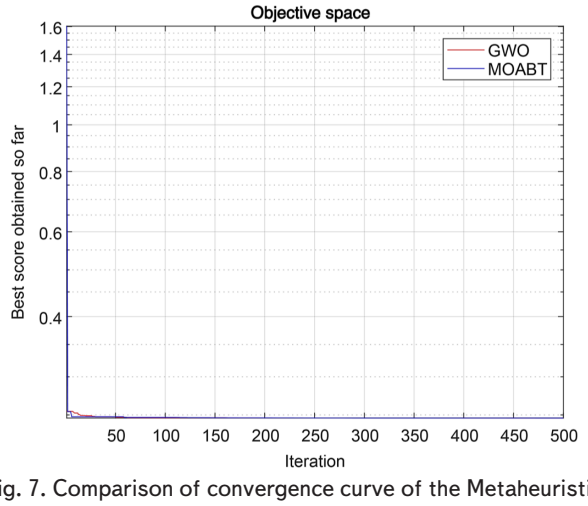


Fig. 7. Comparison of convergence curve of the Metaheuristic optimization algorithm based on the Adams-Bashforth two-step scheme (MOABT) technique and GWO in problem 6

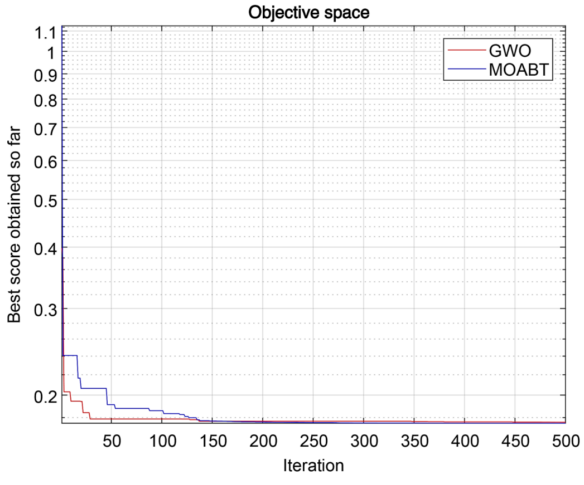


Fig. 8. Comparison of convergence curve of the Metaheuristic optimization algorithm based on the Adams-Bashforth two-step scheme (MOABT) technique and GWO in problem 7

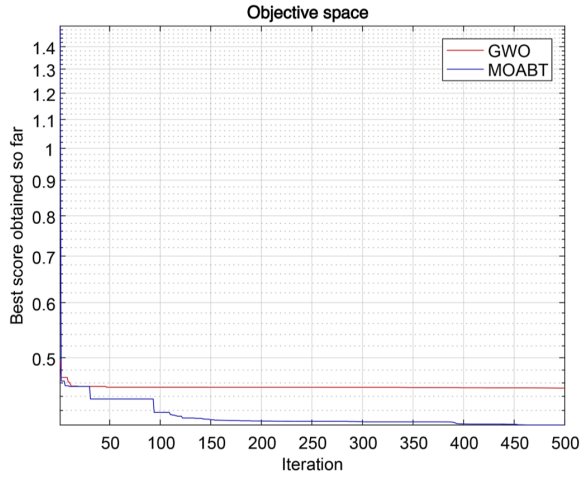


Fig. 9. Comparison of convergence curve of the Metaheuristic optimization algorithm based on the Adams-Bashforth two-step scheme (MOABT) technique and GWO in problem 8

## 6. Discussion of experimental results

The aim of this paper is to find a fast and efficient way to train artificial neural networks to classify data and approximate functions. The metaheuristic algorithm based on the two-step Adam-Bashforth method was used. In Table 1 and Fig. 2–9, the new algorithm outperformed the GWO algorithm in goal convergence by taking 500 iterations for each algorithm.

The new algorithm was compared with the GWO algorithm, and the results of the developed algorithm were more efficient and more accurate than those of the GWO algorithm, as shown in Table 1 and graphs from Fig. 2–9. The best mean square error was obtained for the given problems as found in Table 1 and Fig. 2–9. And we got the global solution for the taken problems.

The scope of the study is to classify the data, approximate the functions, and obtain the least value for the mean square error.

This study is distinguished by its speed in classifying training data and approximate functions.

One of the disadvantages of the method is that it takes time and effort to train artificial neural networks to classify data and approximate functions, and we hope to solve this problem soon.

The difficulties encountered by the researchers in this paper are how to obtain training data to test the proposed method.

## 7. Conclusions

1. The MOABT algorithm is more accurate and more efficient than the GWO algorithm that depends on randomness in training artificial neural networks.

2. The training results were very encouraging for their convergence of the global solution in the classification of data and approximation of functions. The MOABT algorithm is superior to the GWO algorithm in training artificial neural networks.

3. The rate of development of the new algorithm in relation to the wolf algorithm is as follows: in the problem of XOR 65 %, in the problem Balloon 120 %, in the problem Iris 1 %, in the problem Breast cancer 63 %, in the problem Heart 74 %, in the function Sigmoid 1 %, in the function Cosine 1 %, and in the function Sine 13 %.

## Acknowledgments

We extend our sincere thanks and appreciation to the University of Mosul, the College of Computer Science and Mathematics, and the College of Education for Pure Sciences, for their cooperation and support in the completion of this paper.

## References

1. Yilmaz, M., Kayabasi, E., Akbaba, M. (2019). Determination of the effects of operating conditions on the output power of the inverter and the power quality using an artificial neural network. Engineering Science and Technology, an International Journal, 22 (4), 1068–1076. doi: https://doi.org/10.1016/j.jestch.2019.02.006
2. Vahora, S. A., Chauhan, N. C. (2019). Deep neural network model for group activity recognition using contextual relationship. Engineering Science and Technology, an International Journal, 22 (1), 47–54. doi: https://doi.org/10.1016/j.jestch.2018.08.010
3. Maleki, Ghazvini, Ahmadi, Maddah, Shamshirband. (2019). Moisture Estimation in Cabinet Dryers with Thin-Layer Relationships Using a Genetic Algorithm and Neural Network. Mathematics, 7 (11), 1042. doi: https://doi.org/10.3390/math7111042
4. Farzaneh-Gord, M., Mohseni-Gharyehsafa, B., Arabkoohsar, A., Ahmadi, M. H., Sheremet, M. A. (2020). Precise prediction of biogas thermodynamic properties by using ANN algorithm. Renewable Energy, 147, 179–191. doi: https://doi.org/10.1016/j.renene.2019.08.112
5. Basheer, I. A., Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. Journal of Microbiological Methods, 43 (1), 3–31. doi: https://doi.org/10.1016/s0167-7012(00)00201-3

6. Li, J., Cheng, J., Shi, J., Huang, F. (2012). Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement. Advances in Computer Science and Information Engineering, 553–558. doi: https://doi.org/10.1007/978-3-642-30223-7_87
7. Aljarah, I., Faris, H., Mirjalili, S. (2016). Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Computing, 22 (1), 1–15. doi: https://doi.org/10.1007/s00500-016-2442-1
8. Wang, L., Zeng, Y., Chen, T. (2015). Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. Expert Systems with Applications, 42 (2), 855–863. doi: https://doi.org/10.1016/j.eswa.2014.08.018
9. Yang, X. (2010). Nature-Inspired Metaheuristic Algorithms. Luniver Press, 75.
10. Wang, G.-G., Gandomi, A. H., Alavi, A. H., Hao, G.-S. (2013). Hybrid krill herd algorithm with differential evolution for global numerical optimization. Neural Computing and Applications, 25 (2), 297–308. doi: https://doi.org/10.1007/s00521-013-1485-9
11. Reed, R., Marks, R. J. (1999). Neural Smithing. The MIT Press. doi: https://doi.org/10.7551/mitpress/4937.001.0001
12. Dey, N., Ashour, A. S., Bhattacharyya, S. (Eds.) (2020). Applied Nature-Inspired Computing: Algorithms and Case Studies. Springer, 275. doi: https://doi.org/10.1007/978-981-13-9263-4
13. Dey, N. (Ed.) (2018). Advancements in Applied Metaheuristic Computing. IGI Global. doi: https://doi.org/10.4018/978-1-5225-4151-6
14. Gandomi, A. H., Yang, X.-S., Alavi, A. H. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with Computers, 29 (1), 17–35. doi: https://doi.org/10.1007/s00366-011-0241-y
15. Mirjalili, S., Mirjalili, S. M., Lewis, A. (2014). Grey Wolf Optimizer. Advances in Engineering Software, 69, 46–61. doi: https://doi.org/10.1016/j.advengsoft.2013.12.007
16. Mirjalili, S., Lewis, A. (2016). The Whale Optimization Algorithm. Advances in Engineering Software, 95, 51–67. doi: https://doi.org/10.1016/j.advengsoft.2016.01.008
17. Yang, X.-S. (2009). Firefly Algorithms for Multimodal Optimization. Lecture Notes in Computer Science, 169–178. doi: https://doi.org/10.1007/978-3-642-04944-6_14
18. Yang, X., Hossein Gandomi, A. (2012). Bat algorithm: a novel approach for global engineering optimization. Engineering Computations, 29 (5), 464–483. doi: https://doi.org/10.1108/02644401211235834
19. Yu, J., Wang, S., Xi, L. (2008). Evolving artificial neural networks using an improved PSO and DPSO. Neurocomputing, 71 (4-6), 1054–1060. doi: https://doi.org/10.1016/j.neucom.2007.10.013
20. Valian, E., Mohanna, S., Tavakoli, S. (2011). Improved Cuckoo Search Algorithm for Feed forward Neural Network Training. International Journal of Artificial Intelligence & Applications, 2 (3), 36–43. doi: https://doi.org/10.5121/ijaia.2011.2304
21. Jaddi, N. S., Abdullah, S., Hamdan, A. R. (2015). Multi-population cooperative bat algorithm-based optimization of artificial neural network model. Information Sciences, 294, 628–644. doi: https://doi.org/10.1016/j.ins.2014.08.050
22. Mirjalili, S. (2015). How effective is the Grey Wolf optimizer in training multi-layer perceptrons. Applied Intelligence, 43 (1), 150–161. doi: https://doi.org/10.1007/s10489-014-0645-7
23. Nandy, S., Sarkar, P. P., Das, A. (2012). Analysis of a nature inspired firefly algorithm based back-propagation neural network training. arXiv. doi: https://doi.org/10.48550/arXiv.1206.5360
24. Heidari, A. A., Faris, H., Aljarah, I., Mirjalili, S. (2018). An efficient hybrid multilayer perceptron neural network with grasshopper optimization. Soft Computing, 23 (17), 7941–7958. doi: https://doi.org/10.1007/s00500-018-3424-2
25. Xu, J., Yan, F. (2018). Hybrid Nelder–Mead Algorithm and Dragonfly Algorithm for Function Optimization and the Training of a Multilayer Perceptron. Arabian Journal for Science and Engineering, 44 (4), 3473–3487. doi: https://doi.org/10.1007/s13369-018-3536-0
26. Tang, R., Fong, S., Dey, N. (2018). Metaheuristics and Chaos Theory. Chaos Theory. doi: https://doi.org/10.5772/intechopen.72103
27. Karaagac, B. (2019). Two step Adams Bashforth method for time fractional Tricomi equation with non-local and non-singular Kernel. Chaos, Solitons & Fractals, 128, 234–241. doi: https://doi.org/10.1016/j.chaos.2019.08.007
28. Butcher, J. C. (2016). Numerical Methods for Ordinary Differential Equations. John Wiley & Sons. doi: https://doi.org/10.1002/9781119121534
29. Ahmadianfar, I., Heidari, A. A., Gandomi, A. H., Chu, X., Chen, H. (2021). RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method. Expert Systems with Applications, 181, 115079. doi: https://doi.org/10.1016/j.eswa.2021.115079
30. Belew, R. K., McInerney, J., Schraudolph, N. N. (1990). Evolving networks: Using the genetic algorithm with connectionist learning. CSE Technical report #CS90-174. Available at: https://nic.schraudolph.org/pubs/BelMcISch92.pdf