Ahmed A. Abusnaina*, Rosni Abdullah and Ali Kattan

Self-Adaptive Mussels Wandering Optimization Algorithm with Application for Artificial Neural Network Training

https://doi.org/10.1515/jisys-2017-0292
Received June 17, 2017; previously published online February 21, 2018.

Abstract: The mussels wandering optimization (MWO) is a recent population-based metaheuristic optimization algorithm inspired ecologically by mussels' movement behavior. The MWO has been used successfully for solving several optimization problems. This paper proposes an enhanced version of MWO, known as the enhanced-mussels wandering optimization (E-MWO) algorithm. The E-MWO aims to overcome the MWO shortcomings, such as lack in explorative ability and the possibility to fall in premature convergence. In addition, the E-MWO incorporates the self-adaptive feature for setting the value of a sensitive algorithm parameter. Then, it is adapted for supervised training of artificial neural networks, whereas pattern classification of real-world problems is considered. The obtained results indicate that the proposed method is a competitive alternative in terms of classification accuracy and achieve superior results in training time.

Keywords: Mussels wandering optimization, self-adaptive, metaheuristic, neural networks, pattern classification.

1 Introduction

The artificial neural network (ANN) is an interconnected set of nodes (artificial neurons) via a series of adjusted weights. These neurons use a mathematical model for information processing to accomplish a variety of tasks such as identification of objects and patterns, making decisions based on prior knowledge, and prediction of future events based on past experience [9, 15, 37].

The ANN can be applied to determine a nonlinear relationship between a set of features by iterative training of neurons using the obtained data from the environment [4, 25]. The training process of the ANN deals with adjusting the connection weights and/or structure of the network depending on a specific training algorithm [12]. The search space of the ANN connection weights is considered as a continuous optimization problem because it is high dimensional and multimodal; also, it could be corrupted by noises or missing data [22, 28].

The supervised training of the ANN has been tackled by two main paradigms: gradient descent (GD) and population-based metaheuristic (P-Metaheuristic) algorithms. The GD paradigm uses the error gradient to descend the error surface, such as the back-propagation (BP) and Levenberg–Marquardt (LM) algorithms. The derivative of the error function is computed in order to adjust the network weights. The GD suffers from convergence slowness and high possibility to fall into local minima [24, 45]. On the other hand, the algorithms in the P-Metaheuristic can work on different and multiple regions of the solution space for the same problem simultaneously via a set of individuals [13]. The P-Metaheuristic might be more efficient by concerning the exploration ability of the whole search space and obtaining an acceptable solution [19, 39]. However,

***Corresponding author: Ahmed A. Abusnaina,** Department of Computer Science, Faculty of Engineering and Technology, Birzeit University, Birzeit, Ramallah, Palestine, e-mail: aabusnaina@birzeit.edu
Rosni Abdullah: School of Computer Sciences, Universiti Sains Malaysia, 11800, Penang, Malaysia
Ali Kattan: IT Department, Ishik University, Qazi Muhammad, Erbil, Iraq

the P-Metaheuristic algorithms might have higher computational cost and more complex structures [19, 47], whereas the performance often depends on the algorithm settings and the problem characteristics [31].

The mussels wandering optimization (MWO) algorithm is a recent and novel P-Metaheuristic algorithm, inspired ecologically for global optimizations by An et al. [6]. The MWO models the mussels' movement behavior when they form a bed pattern in their surrounding habitat to solve complex optimization problems. The MWO depends on the stochastic decision and Levy walk in order to find the optimal solution. The Levy walk is efficient, provides faster diffusion, and prevents revisiting the same sites [14]. The MWO maintains information about the previous useful solutions, unlike other algorithms [e.g. the genetic algorithm (GA)] that destroys the previous knowledge of the solution [53]. Also, the MWO uses primitive mathematics, and it does not have a special kind of operators (e.g. mutation and crossover). In addition, the MWO has the advantage of working in parallel on a multiple set of solutions, as this feature is common for the P-Metaheuristic algorithms. It also inspires the cooperation, competition, and information sharing among the mussel population, which enhance the ability of searching for the global optimal solution.

In this paper, the MWO is analyzed, and its pros and cons are discussed. An enhanced version of the MWO (specifically the E-MWO algorithm) is proposed to overcome the MWO shortcomings. The E-MWO is then adapted for supervised training of the ANN to perform pattern classification. The proposed method aims to minimize the ANN training time and achieve better classification accuracy. These objectives are validated by making use of several real-world benchmark classification problems.

The rest of this paper is organized as follows: Section 2 presents the related work on the ANN training methods. Section 3 describes the MWO algorithm. Section 4 presents the proposed E-MWO algorithm. In Section 5, the adaptation of the MWO-based algorithms for training the ANN is demonstrated. Section 6 presents the experimental setup, and Section 7 discusses the obtained results. Finally, Section 8 concludes the paper.

2 Related Work

The P-Metaheuristic algorithms are inspired from various aspects in the real world. Many of these algorithms are inspired from the biological processes in the living creatures or from social interactions among animals. Several P-Metaheuristic algorithms have been employed for supervised training of the ANN; however, this section will focus on the rival methods.

The GA has been used for training the ANN in three different approaches. The first one involves optimizing the weights of the ANN with a fixed structure [16, 18, 42, 43]. Some of these methods were evaluated based on using simple logic problems such as XOR, encode–decode, and parity problem. The second approach is using the GA for constructing a suitable ANN structure [44]. The last approach is employing the GA by evolving both structure and weights of the ANN [12, 38].

The harmony search algorithm (HSA), inspired from the improvisation process of musicians, was adapted for training the ANN by Kattan and Abdullah [31], where both the sum squared error (SSE) and classification error percentage (CEP) error calculation methods are used as fitness function. The training process was terminated by reaching the maximum number of improvisations (i.e. iterations), and only four datasets are considered in their study. A new variant of HSA that has the property of adaptive setting for a sensitive HSA parameter by utilizing the ratio of best-to-worst harmony (HS-BtW algorithm) was proposed by Kattan et al. [30, 32]. The HS-BtW was adapted for training the ANN, whereas the termination criterion depends mainly on the quality measure of the solutions and number of improvisations. Different variations of the HSA were also used for training the ANN such as the HSA, self-adaptive global best HS, and improvised harmony search (IHS) by Kulluk et al. [34, 35].

Both the ANN weights and structure were evolved by an approach called ESPNet [53]. ESPNet employed the standard particle swarm optimization (PSO) and discrete PSO (DPSO). The network structure dimension is represented by bits and manipulated by the DPSO, while the connection weight dimension is represented by real values and manipulated by the PSO, whereas individuals are represented by a matrix–hierarchy

structure. Only two datasets were used for evaluation, and they were not compared against other training methods. The ESPNet results were good in generalization ability; however, additional cost in terms of computational time was incurred. The Multi-dimensional PSO (MDPSO) was used for training the ANN by evolving its structure rather than weights [33]. The mean squared error (MSE) was used as a fitness function, whereas the 10-bit parity and another three classification problems were used for evaluation. The results of the MDPSO were compared against other variants of the PSO, GA, and BP. The MDPSO results were argued to be superior in terms of generalization ability; however, further improvements were suggested in terms of speed and accuracy. Different topologies of the PSO individuals are used for the purpose of training the ANN, which are proposed in Refs. [41] and [49].

Other P-Metaheuristic algorithms are still employed for ANN training, such as the artificial bee colony (ABC) [11, 28], ant colony optimization (ACO) [10, 40, 46, 50], and group search optimizer (GSO) [22, 23]. Paradigms other than the GD and P-Metaheuristic exist in literature with the aim of training the ANN. Hybrid methods that combine the GD and P-Metaheuristic have been proposed to improve upon the GD algorithms, such as the GA with BP [5] and PSO with BP [51]. However, the advantages of those methods are arguable [11]. Algorithms that are based on single-solution metaheuristic are also used, such as simulated annealing [36].

The P-Metaheuristic algorithms are employed for supervised training of the ANN, to overcome the drawbacks of the GD learning algorithms such as slowness, local minima, necessity of using differentiable activation function, and training oscillation [28, 30]. However, the P-Metaheuristic algorithms still suffer from high computational cost, complex structures, and sensitivity to parameter settings, which might lead to different performance measurements.

3 Mussels Wandering Optimization Algorithm

The MWO is essentially an algorithm that applies an optimization, stochastic, and metaheuristic process seeking to find the best solution for a given problem [6]. Starting by a randomized population of candidate solutions, an iterative process is conducted until a certain measure is reached.

Through the MWO algorithm, the population of mussels consists of N individuals. These individuals are in a certain spatial region of marine bed called the habitat. The habitat is mapped to a d-dimensional space S^d of the problem to be optimized, where the objective function value $f(s)$ at each point $s \in S^d$ represents the nutrition provided by the habitat. Each mussel has a position $x_i := (x_{i1}, \ldots, x_{id}); i \in N$, which therefore forms a specified spatial bed pattern.

The MWO algorithm is initialized with a population of random candidate solutions. Each mussel is assigned a randomized position. Then, it iteratively moves through the solution space. The mussel population is attracted toward the location of the global-best mussel (the global-best mussel is the mussel that has the best fitness value) achieved so far across the whole population. The MWO algorithm is composed of six main steps, which are as follows: (1) Initialize the mussel population and the algorithm parameters. (2) Calculate the Euclidean distance (D_{ij}), range references (r_s and r_l), and range densities (ζ_s and ζ_l) for each mussel. (3) Determine the movement probability (P_i) for each mussel. (4) Update the position of the mussels. (5) Evaluate the fitness of each mussel after position updating and perform the global-best selection scheme. (6) Examine the termination criteria; if it is not satisfied, then go back to step 2. Figure 1 shows the flowchart of the MWO, where the detailed MWO is given in Algorithm 1. The set of equations that is used by the MWO is demonstrated in Table 1.

The MWO algorithm has been used to solve nonlinear unimodal and multimodal benchmark functions [6]. Most of the multimodal functions have many local optima, and these optima exponentially increase with the dimension space. A new clustering algorithm by combining the K-means clustering method and the MWO is currently proposed by Yan et al. [52]. Tests on six standard datasets are performed, whereas the results demonstrate the validity and superiority of their proposed method over some representative clustering ones. Also, the K-MWO is used for clustering the time-sharing characteristic of the household energy consumption [7].

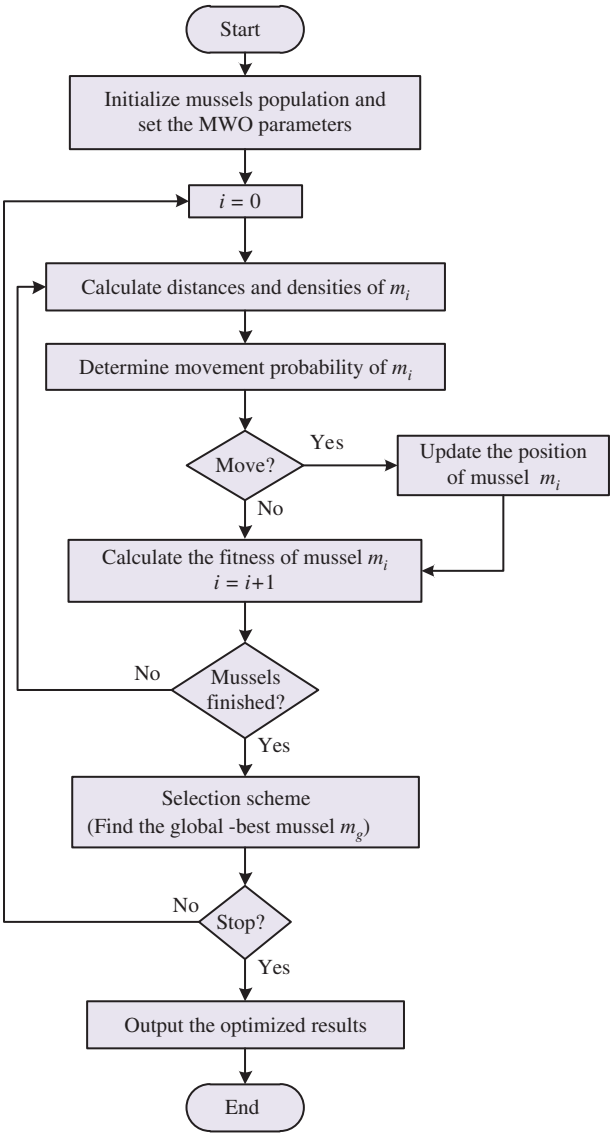

Figure 1: MWO Flowchart.

The MWO algorithm showed the ability of training the ANN in considerably low convergence time, while the classification accuracy was mostly at par with other rival training methods. However, it was noticed that the MWO has some shortcomings that limit its performance [1–3]. The MWO might terminate prematurely due to the high selective pressure on the global-best mussel. The coefficients and parameters of the MWO must be determined statically before the run time. However, some of these parameters are sensitive to the optimization problem, and they highly affect the algorithm performance, especially the shape parameter (μ) (see Refs. [2] and [6]). The MWO depends on the single-step approach to update the mussel position during the searching process for the best place. The use of this single-step approach shows the lack in the explorative ability of the MWO. The MWO depends on the number of iterations as a termination condition. Using this condition, even if the candidate solutions are not good enough, the MWO will continue its run until it reaches the whole number of iterations. This behavior might incur unnecessary iterations that will not lead to a better solution. An initial idea for enhancing the MWO was proposed by Abusnaina et al. [2] to achieve better performance.

Algorithm 1: The MWO Algorithm.

1: **Initialization:**
2: Set $t=0$;
3: **For** (mussel m_i, $i=1$ to N) **do**
4: Uniformly randomize the initial position $x_i(0)$ for each mussel m_i from the range $[x_{min}, x_{max}]$
5: Calculate the initial fitness value of the mussel $f(x_i(0))$
6: **end for**
7: Find the global-best mussel m_g and record its position as x_g
8: **Iteration:**
9: **while** ($t \leq MaxIterations$) **do**
10: **for** (mussel m_i, $i=1$ to N) **do**
11: Calculate the distances D_{ij} from m_i to all other mussels by Eq. (A.1);
12: Calculate short-range reference $r_s(t)$ and long-range reference r_l by Eq. (A.2);
13: Calculate short-range density ζ_s and long-range density ζ_l by Eq. (A.3) and Eq. (A.4);
14: Compute the moving probability $p_i(t)$ according to Eq. (A.5);
15: **If** ($P_i(t)=1$) **then**
16: Generate step length $\ell_i(t)$ by Eq. (A.6)
17: **else**
18: $\ell_i(t)=0$
19: **end if**
20: Update the mussel position coordinate $\dot{x}_i(t)$ using Eq. (A.7) according to mussel m_g
21: Calculate the new fitness value after the mussel update its position coordinate $f(\dot{x}_i(t))$
22: **end for**
23: Rank all mussels by their fitness and Find the global-best mussel (m_g) and update the best position x_g
24: Set $t=t+1$;
25: **end while**
26: Output the optimized results and end the algorithm

Table 1: List of Equations Used in MWO Algorithm [6].

Eq. (A.1):	$D_{ij} := \| x_i - x_j \| = \left[\sum_{k=1}^{d} (x_{ik} - x_{jk})^2\right]^{\frac{1}{2}}$	D_{ij}: spatial distance between mussels m_i and m_j in S^d N: number of mussels. $i, j \in N$
Eq. (A.2):	$\begin{cases} r_s(t) := \alpha.max_{i,j\in N}\{D_{ij}(t)\}/\delta \\ r_l(t) := \beta.max_{i,j\in N}\{D_{ij}(t)\}/\delta \end{cases}$	r_s: short-range reference. r_l: long-range reference. α, β are positive coefficients with $\alpha<\beta$. $\max_{i,j\in N}\{D_{ij}(t)\}$: is the maximum distance among all mussels at iteration t. δ: scale factor of space
Eq. (A.3):	$\zeta_{si} = \#(D_i < r_s)/(r_s N)$	ζ_{si}: short-range density, ζ_{li}: long-range density, where $\#(A<b)$ is used to compute the count in set A satisfying $a<b$; $a \in A$. D_i is the distance matrix from mussel m_i to other mussels
Eq. (A.4):	$\zeta_{li} = \#(D_i < r_l)/(r_l N)$	
Eq. (A.5):	$P_i := \begin{cases} 1 & \text{if } a - b\,\zeta_{si} + c\,\zeta_{li} > z \\ 0 & \text{otherwise} \end{cases}$	a, b, and c are positive constant coefficients. z: is a value randomly sampled from the uniform distribution [0,1]
Eq. (A.6):	$\ell_i = \gamma[1 - rand()]^{-1/(\mu-1)}$	ℓ_i: step length, μ: is the shape parameter, which is known as the Levy exponent; $1<\mu<3$. γ: the walk scale factor
Eq. (A.7):	$\dot{x}_i := \begin{cases} x_i + \ell_i \Delta_g & \text{if } P_i = 1 \\ x_i & \text{if } P_i = 0 \end{cases}$	$\dot{x}_i$: the new mussel-position coordinate. x_i: the current mussel-position coordinate. x_g: the position coordinate of the global-best mussel (m_g). $\Delta_g = x_i - x_g$

4 Enhanced-Mussels Wandering Optimization Algorithm

The E-MWO algorithm is proposed to overcome the shortcomings of the MWO in order to solve the optimization problems more efficiently, especially improving the classification accuracy of the ANN. A new hybrid-selection scheme is introduced to cope with the premature convergence problem. The value of the sensitive

parameter μ is set dynamically and adaptively depending on the quality of the candidate solutions. The multi-step length approach is used in order to make the algorithm more explorative to the solution space. Another improvement lies in terminating the E-MWO algorithm depending on the dynamic quality of the solutions instead of the iteration number only.

4.1 Hybrid-Selection Scheme

The original MWO algorithm uses the global-best mussel as guidance to update the position of all other mussels [see Eq. (A.7) in Table 1]; however, this selection scheme is good at the early iterations of the optimization process. Typically, the fitness of the global-best mussel (m_g) becomes steady for a large number of iterations at the late stage of the optimization process. The *SteadyState* means the fitness value of the global-best mussel (f_{m_g}) does not change; the mussel is stuck at the same position, and no further improvement to the solution could be produced. This phenomenon can be referred to as the problem of premature convergence [17].

This problem is solved by the E-MWO by hybridizing two selection schemes simultaneously: the global-best and random selection schemes. At any iteration t, the mussel population will follow the global-best mussel (m_g) in updating their positions as long as the *SteadyState* is not detected. However, if the *SteadyState* is detected as demonstrated in Eq. (2), the fitness value of the global-best mussel does not improve for the last T of iterations. Another mussel is selected randomly from the *best-M* of mussels. The randomly selected mussel will be used as a guidance mussel. Through the E-MWO, the guidance selected mussel (m_s), which is followed by other mussels during the position update, is either the global-best mussel or the randomly selected mussel; thus, new regions of solution space could be explored.

$$\psi(t)=\begin{cases}1 & f_{m_g}(t)=f_{m_g}(t-1)\\ 0 & \text{Otherwise}\end{cases} \tag{1}$$

$$StdState=\begin{cases}1 & \left(\sum_{t=\hat{t}}^{t=\hat{t}-T}\psi(t)\right)=T\\ 0 & \text{Otherwise}\end{cases} \tag{2}$$

where the steady state is detected if $stdState=1$, $f_{m_g}(t)$ is the fitness value of the global-best mussel at iteration t, T is the number of iterations to detect the *SteadyState*, and $\hat{t}$ is the current iteration number.

The *best-M* of the mussels is a subset from the whole mussel population; the *best-M* $\subset$ mussels population. In order to determine the *best-M* of the mussels, sorting the mussel population is performed based on their fitness values. Appendix A illustrates an example of how the *best-M* is determined. The following line numbers in Algorithm 2 includes the proposed hybrid-selection scheme: 3, 8, 18, 27, 28, 29, 30, and 31. The formula of updating the mussel position [i.e. Eq. (A.7) in Table 1] used by the MWO is rewritten in Eq. (3) and is used by the E-MWO in order to adapt the new modifications that are made to the selection scheme.

$$\hat{x}_i=\begin{cases}x_i+\ell_{ik}\Delta_s & P_i(t)=1\\ x_i & P_i(t)=0\end{cases} \tag{3}$$

where $\hat{x}_i$ is the new mussel-position coordinate, x_i is the current mussel-position coordinate, and x_s is the position coordinate of the selected-mussel (m_s). $\Delta_s=x_i-x_s$.

4.2 Self-Adaptive Setting of Shape Parameter (μ)

Achieving the best performance for the MWO depends on choosing the proper value of μ [1, 2, 6]. However, finding this proper value needs conducting many empirical experiments by trail-and-error. The μ value is important for the MWO as it is used to determine the extent of mussel movement, as the mussel move from its

Algorithm 2: The E-MWO Algorithm.

1: **Initialization:**
2: Set $t=0$;
3: Define *SteadyState* = the fitness value of the global-best mussel $(f_{(m_g)})$ is the same for the last ***T*** iterations
4: **for** (mussel m_i, $i=1$ to N) **do**
5: Uniformly randomize the initial position $x_i(0)$ for each mussel m_i from the range $[x_{min}, x_{max}]$
6: Calculate the initial fitness value of the mussel $f(m_i(0))$
7: **end for**
8: Find the global-best mussel (m_g), record its position, and set it as selected mussel m_s
9: **Iteration:**
10: **while** ($t \leq$ *MaxIterations* **AND** $U_R(t) \geq \epsilon_1$ **AND** $S_R(t) \leq \epsilon_2$) **do**
11: **for** (mussel m_i, $i=1$ to N) **do**
12: Calculate the distances D_{ij} from m_i to all other mussels by Eq. (A.1);
13: Calculate short-range reference $r_s(t)$ and long-range reference $r_l(t)$ by Eq. (A.2);
14: Calculate short-range density ζ_s and long-range density ζ_l by Eq. (A.3) and Eq. (A.4);
15: Compute the moving probability $P_i(t)$ according to Eq. (A.5);
16: **if** ($P_i(t)=1$) **then**
17: Generate all steps length $\ell_{i0}(t)$ to $\ell_{ik}(t)$ by Eq. (A.6)
18: Update the mussel position coordinate $\hat{x}_i(t)$ using Eq. (3) according to m_s
19: Calculate the new fitness value after the mussel update its position coordinate $f(\hat{x}_i(t))$
20: **else**
21: Set all steps length $\ell_{i0}(t)$ to $\ell_{ik}(t)$ to 0
22: **end if**
23: **end for**
24: Calculate the $iS_R(t)$ by Eq. (5) and $S_R(t)$ by Eq. (6)
25: Calculate the $U_R(t)$ by Eq. (8)
26: Calculate the new value of shape parameter $\mu(t)$ by Eq. (9)
27: Rank all mussels by their fitness and Find the global-best mussel and set it as selected mussel m_s
28: Check the *SteadyState*
29: **if** (*SteadyState* is detected) **then**
30: select a mussel randomly from the ***best-M*** of the mussels population and set it as selected mussel m_s
31: **end if**
32: Set $t=t+1$;
33: **end while**
34: Output the optimized results and end the algorithm

current position to another toward the global-best mussel [see Eq. (A.6) in Table 1]. Adding the self-adaptive feature to the E-MWO will allow setting the value of μ dynamically and adaptively depending on newly introduced quality measurements: the similarity ratio (S_R) and update ratio (U_R), as shown in Algorithm 2 through Lines 24, 25, and 26.

The instantaneous similarity ratio (iS_R) is calculated at each iteration. It is the ratio of the number of mussels that have the same fitness value at a certain iteration t divided by the population size of the mussels. The iS_R has a fluctuation value, and it does not provide sufficient information on how much the solution has converged. More discussion will be provided in the Results section. Therefore, a more accurate and stable measure is desired. The overall similarity ratio (S_R) is used, which returns the accumulative average of the iS_R over the number of iterations consumed to reach the current solution; the iS_R and the S_R are given in Eqs. (5) and (6), respectively. Appendix B illustrates a numerical example for calculating the iS_R and the S_R.

$$\varphi(i)=\begin{cases}1 & f(m_i)=f(m_j) \quad i\neq j\\ 0 & \text{Otherwise}\end{cases} \tag{4}$$

$$iS_R(t)=\frac{\sum_{i=1}^{N}\varphi(i)}{N} \tag{5}$$

$$S_R(t) = \frac{\sum_{t=1}^{t=\grave{t}} iS_R}{t} \tag{6}$$

where $f(m_i)$ is the fitness value of the mussel m_i at iteration t, N is the mussel population size, and $\grave{t}$ is the current iteration number.

The U_R provides information about the dynamicity of the population. It returns the ratio of the number of mussels that update their positions and move to a new place at a certain iteration t over the mussel population size. The U_R formula is given in Eq. (8).

When μ is small, i.e. $1.0 < \mu \leq 1.4$, or μ is large, i.e. $2.0 \leq \mu < 3.0$, the performance of the MWO is weak, e.g. the trained ANN has poor classification accuracy. In addition, it is founded empirically that the U_R becomes very small and the S_R becomes very high in these ranges of μ [2]. A low value of U_R and a high value of S_R means that the mussel population is stuck almost at the same place. In other words, the diversity of the population is very low, and the MWO algorithm seems to fall in the premature convergence. The property of setting the value of μ adaptively and dynamically is necessary, therefore, keeping the U_R as high as possible and the S_R as small as possible. Also, setting the μ adaptively is vital to avoid performing tedious experiments for finding the proper value of μ. Equation (9) demonstrates the calculation of the dynamic and self-adaptive μ value at any iteration t.

$$\phi(i) = \begin{cases} 1 & \text{if } m_i \text{ update its position} \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

$$U_R(t) = \frac{\sum_{i=1}^{N} \phi(i)}{N} \tag{8}$$

$$\mu(t) = \mu_c + \lambda_1 \, S_R(t) + \lambda_2 \, U_R(t) \tag{9}$$

where N is the mussel population size, μ_c is the shape parameter constant, and λ_1 and λ_2 are coefficients.

4.3 Multi-Step Length Approach

The step length (ℓ) is used in the MWO to update the mussel position [see Eqs. (A.6) and (A.7) in Table 1]. The original MWO uses a one-step length to update the mussel-position coordinate, i.e. all coordinates changed with the same length extent. Using a single-step length might cause a lack in the explorative ability of the MWO. The E-MWO uses a multi-step length rather than once. Each category of the mussel-position coordinate is assigned a separate step length.

Utilizing the multi-step length approach will improve the explorative ability of the mussel population to the solution space using sufficient randomness into the step length extent. The increase in the explorative ability will enhance the diversity of the mussel population [54]. However, maintaining the population in a balanced degree of diversity is essential to ensure that the solution space is adequately searched [21]. The multi-step length approach is proposed in Algorithm 2, in Lines 17, 18, and 21.

For instance, if the E-MWO is used to train a four-layer ANN, then each mussel (m_i) will use a three-step length; the number of steps is equal to the number of ANN layers minus one as follows: ℓ_{i1}: $Input - Hidden_1$, ℓ_{i2}: $Hidden_1 - Hidden_2$, ℓ_{i3}: $Hidden_2 - Output$.

4.4 Dynamic Termination Criterion

The termination criterion for the E-MWO depends mainly on the dynamic quality measurements of the candidate solutions. The calculation of these measures is performed repetitively at each iteration. The quality measure includes two newly introduced measurements: the U_R and the S_R. These two measures are utilized to guide the E-MWO as to when to terminate its process. The S_R and the U_R are the same variables used in

setting the proper value of μ, which were given earlier in Eqs. (6) and (8), respectively. At any iteration t, the $U_R(t)$ should not be less than ϵ_1, while the $S_R(t)$ should not exceed ϵ_2. As the inequality relations shown below are preserved as the candidate solutions are properly converged to the optimal solution, accordingly, better performance is supposed to be achieved (Algorithm 2 at Line 10).

$$U_R(t) \geq \epsilon_1 \textbf{ AND } S_R(t) \leq \epsilon_2,$$

where ϵ_1 and ϵ_2 are constant values determined experimentally.

However, the *MaxIterations* is still used as an auxiliary condition to stop the E-MWO in order to limit the number of iterations if the quality measures are unable to trigger the termination of the algorithm. Furthermore, using the *MaxIterations* condition will guarantee that the best solution will be produced at a reasonable amount of time.

The aforementioned new parts and the proper modifications made over the original MWO is combined and form the E-MWO algorithm as shown by Algorithm 2. Equations (A.1) and (A.6) used by the E-MWO are the same equations used by the original MWO, which are presented in Table 1, while the other equations are already defined and explained earlier throughout this section.

It is worth mentioning that the update of the mussel position and the calculation of the new fitness are performed only if the movement probability is $P_i(t) = 1$, through Lines 18 and 19 in Algorithm 2. While the MWO always performs these operations regardless of the value of $P_i(t)$ (Lines 20 and 21 in Algorithm 1), this would incur unnecessary calculations for these operations if $P_i(t) = 0$. Definitely, the transfer of performing the position update and fitness calculation from Lines 20 and 21 in Algorithm 1 to inside the IF statement through Lines 18 and 19 in Algorithm 2 would lessen the computational time and avoid unnecessary calculations.

5 Adaptation of the E-MWO Algorithm for ANN Training

The adaptation process of the E-MWO for training the ANN has the following aspects that will be addressed and discussed: How can the neural network be represented? How is the quality of the candidate solutions (fitness) during the training process measured? When should the training process be stopped? The E-MWO is an optimization algorithm, which will be employed for finding and adjusting a set of suitable weights and biases for a fixed structure of the ANN.

5.1 ANN Representation

In this research, a three-layer architecture of the feed-forward ANN is considered. Such architecture is shown in Figure 2, where every neuron in layer k is fully connected to all the neurons in the next (forward)

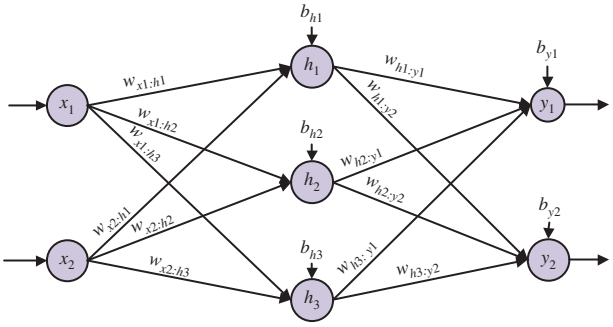

Figure 2: Feed-Forward ANN.

layer $k+1$. The bipolar-sigmoid activation function is used to represent the neurons because it is smooth and binds the output between [–1, 1] [29].

The vector scheme is utilized in this work to represent the feed-forward ANN because it seems more suitable and occupies less memory. Accordingly, each ANN is represented by a vector of connection weights and biases $(\vec{w})$. The network shown in Figure 2 is represented by a weights vector $\vec{w}$ as follows:

$$\vec{w} = [w_{x1h1}, w_{x1h2}, w_{x1h3}, w_{x2h1}, w_{x2h2}, w_{x2h3}, b_{h1}, b_{h2}, b_{h3}, w_{h1v1}, w_{h1v2}, w_{h2v1}, w_{h2v2}, w_{h3v1}, w_{h3v2}, b_{v1}, b_{v2}].$$

Each member in the mussel population represents a complete ANN as demonstrated in Eq. (10). The total number (d) of weights and biases of the network is mapped to the d-dimensional space of the mussel position.

$$m_i = \vec{w}_i = [W_{I-H}, Bias_H, W_{H-O}, Bias_O] \tag{10}$$

5.2 Fitness Measure

The fitness measure is the objective function that quantifies the optimality of the candidate solution (mussel) and evaluates the mussel efficiency. The SSE described in Eq. (11) is used as the mussel fitness value. The SSE measures the difference between the actual output of the network and the desired output of the network. The E-MWO is used as a minimization problem solver; thus, the mussel that has lower SSE value (higher fitness) is the dominant mussel.

$$f(m_i) = \text{SSE} = \sum_{p=1}^{NP} \sum_{n=1}^{NO} (d_p^n - y_p^n)^2 \tag{11}$$

where $f(m_i)$ is the fitness value of the mussel m_i, NP is the number of patterns in the training set, NO is the number of output neurons at the output layer, d_p^n is the desired n^{th} output of the p^{th} pattern, y_p^n is the actual n^{th} output of the p^{th} pattern, and m_i is the mussel individual, i.e. the represented ANN network.

5.3 Termination Condition

Several termination conditions could be used to trigger the training method when to stop its process. Such termination conditions that could be used in adapting the original MWO are (1) accomplishing the maximum number of iterations, (2) the best-mussel fitness satisfies a predefined precision value [e.g. $f(m_g) = 25$], (3) a combination from the previous conditions, e.g. the training stops either when the maximum number of iterations is accomplished or a predefined precision is satisfied. In this research, accomplishing the maximum number of iterations (*MaxIterations*) is used as a termination condition for the MWO because it is more suitable, and it is commonly used in training methods. However, the E-MWO uses the proposed dynamic termination condition, which is demonstrated earlier in Section 4.

6 Settings and Experimental Setup

6.1 Evaluation Method

The merits of the proposed method are demonstrated and validated empirically using a set of benchmarking problems, and many comparisons are conducted against other recent and common rival training methods. The selected methods are the BP from the category of the GD paradigm, GA that was proposed by Dorsey et al. [16], recent algorithms HS-BtW by Kattan et al. [30–32], and PSO [20, 53] from the category of the P-Metaheuristic paradigm.

Two criteria are considered in the performance evaluation: classification accuracy and overall training convergence time. The classification accuracy illustrated in Eq. (12) is considered as the main criterion because of its importance. The classification accuracy is calculated over the testing set, while the training convergence time is reported for the training set. Each training method is evaluated against each benchmarking problem 20 times in order to avoid the randomness factor. Then, the mean value is calculated, and the best out of 20 values is reported for each training method. This method of reporting the results is more fair than reporting the best value only; also, this method is used in the literature such as [20, 26]

$$\text{Accuracy} = \frac{\text{Correctly classified patterns}}{\text{Total no. of patterns}} \times 100\% \tag{12}$$

The statistical two-tailed *t*-test with null hypothesis is used to determine if the results of the proposed method is significantly different from the rival training methods. The *t*-test is the commonly used statistical test to determine the significant differences between the learning algorithms [12, 48]. The null hypothesis is rejected if the *P* value of the *t*-test is smaller than α (i.e. $P \leq \alpha$), whereas α is set to 0.05. As the *P* value becomes smaller, the difference is more significant.

The Java programming language is used in this work to implement all of the ANN modules and all of the proposed and rival training methods. All experimental sessions are conducted independently on the same computer with a 2.2-GHz Intel Core 2 Duo processor and 2 GB of main memory under Microsoft Windows 7 operating system environment.

6.2 The E-MWO Parameters

Different selection of values for the set of the MWO/E-MWO parameters would affect its performance. These parameters and coefficients (N, α, β, δ, a, b, c, and γ) are set as proposed by Ref. [22] and used in Refs. [1] and [2], and they are the same for all datasets as given in Table 2. The setting of the *MaxIterations*, μ_c, λ_1, λ_2, ϵ_1, ϵ_2, T, and M, are based on empirical experiments conducted to select the proper values, which are summarized also in Table 2.

Table 2: Parameter Setting of the E-MWO Algorithm.

Parameter	Symbol	Value
Number of mussels (population size)	N	50
Short-range reference coefficient	α	1.1
Long-range reference coefficient	β	7.5
Space-scale factor	δ	25.5
Moving coefficients	a	0.95
	b	1.26
	c	1.05
Walk-scale factor	γ	0.1
Shape parameter constant	μ_c	1.1
Similarity ratio coefficient	λ_1	−0.3
Update ratio coefficient	λ_2	0.5
Update ratio threshold	ϵ_1	0.2
Similarity ratio threshold	ϵ_2	0.35
Number of iterations to detect the *SteadyState*	T	4
Random selection range	M	70%
Number of iterations	*MaxIterations*	1000
Random initialization range of positions	$[x_{min}, x_{max}]$	[−0.77, 0.77]

Table 3: Benchmark Classification Problems.

Problem	No. of patterns		ANN structure	Dimension (size of $\vec{w}$)	No. of features	No. of classes
	Training	Testing				
Haberman	244	62	3-4-2	26	3	2
Iris	120	30	4-5-3	43	4	3
Magic	7608	1902	10-4-2	54	10	2
Diabetes	614	154	8-7-2	79	8	2
Cancer	546	137	9-8-2	98	9	2
Ionosphere	281	70	33-4-2	146	33	2
Glass	171	43	9-12-6	198	9	6
Thyroid	5760	1440	21-15-3	378	21	3

6.3 Benchmark Problems

The proposed method of the ANN training is validated by making use of widely used benchmark classification problems. The datasets are obtained from the UCI Machine Learning Repository [8]. The selection of the dataset was based on using different datasets from various fields that have different complexities without focusing on a particular type of data in order to prove the generality of the proposed method. The input features of the patterns are normalized to the range [–1, 1] using the Min-Max normalization, so that the bipolar-sigmoid activation function can be effectively applied. In addition, the Min-Max normalization has the advantage of preserving exactly all the relationships among the inputs in the data [27]. After data normalization, each dataset is partitioned into two sets by 80:20. Whereas 80% of the patterns are used for training the network, the remaining 20% out of the patterns are used for testing the generalization ability of the trained network. The detailed specifications of the used benchmark classification problems are given in Table 3.

7 Results

7.1 The E-MWO Dynamics

The E-MWO sets the value of μ adaptively depending on the dynamic quality of the candidate solutions to maintain the U_R as large as possible to ensure the exploration behavior of the algorithm, as can be clearly seen in Figure 3. In addition, the value of μ is changed to reflect the changes in the accumulative S_R, in order to keep the S_R at a suitable range. Because the value of iS_R change irregularly and to avoid the catastrophic change in the μ value, the accumulative S_R is used rather than the instantaneous iS_R. The S_R gives a picture about how much the candidate solutions are close to each other. A high value of S_R means that the algorithm tends to exploit the already visited positions of the global-best mussel rather than explore new positions. The adaptability of μ successfully ensures that the E-MWO makes a balance between the exploration and exploitation behaviors. However, at the late iterations, the E-MWO is unable to make such balance, and the termination condition is satisfied.

The standard MWO uses the global-best scheme to guide other mussels. This scheme increases the selective pressure on the global best. Also, the balance between the exploitation and exploration might be avoided, and the problem of premature convergence may occur. The proposed hybrid-selection scheme is used in the E-MWO to overcome the previous drawback. This modification can be seen in the fitness-convergence graph of the guidance-selected mussel (m_s) in Figure 4. Note that the fitness-convergence graph is plotted for each benchmarking problem; this graph is selected for one session randomly out of the 20 sessions. At the early stage of iterations, the E-MWO uses the global-best scheme. If the steady state is detected, a random mussel is selected. The effect of the random selection scheme can be seen from the abrupt and sharp increase in

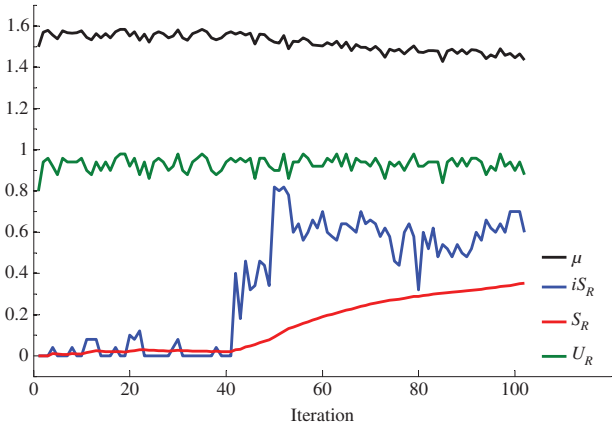

Figure 3: The Graph of μ with iS_R, S_R, and U_R.

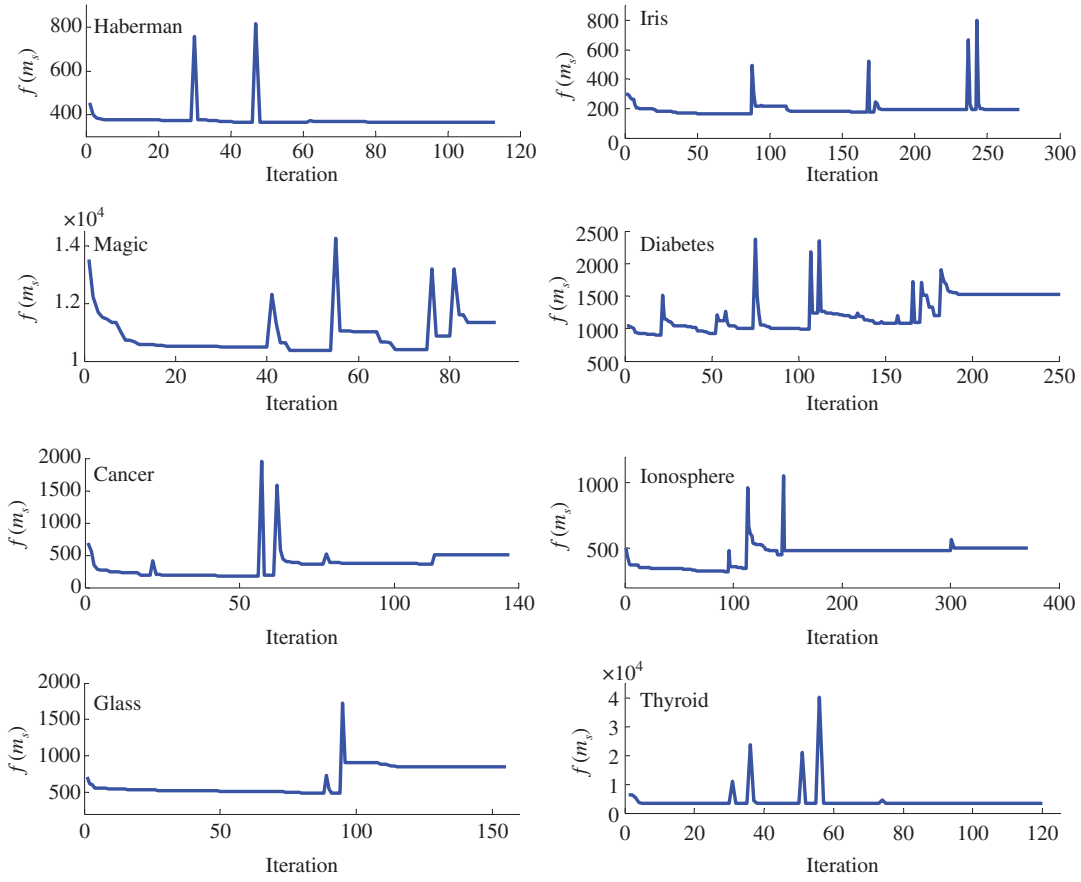

Figure 4: The Fitness Convergence of the Selected Mussel (m_s).

the fitness-convergence graphs. The fitness in Haberman and Thyroid was enhanced slightly after each time the random selection is performed. These problems represent the smallest and largest problem size, i.e. the weight vector $\vec{w}$ size, respectively. The fitness in Iris, Magic, and Diabetes takes more number of iterations to enhance the fitness value after each time the random selection is performed. The fitness convergence in Cancer is unstable; sometime, it takes the benefit of the random selection and the fitness enhanced slightly, and sometimes the fitness becomes far from the optimal.

The fitness in Ionosphere and Glass does not enhance if the random selection is performed. These classification problems are characterized by a large number of features (i.e. Ionosphere has 33 input features) or by multi-class (large number of outputs; i.e. Glass has six classes). Usually, the frequency in using the random selection in these problems is less than the others, as it was used just twice, whereas it was used three times or more in other problems (e.g. Cancer and Diabetes).

A remark should be highlighted: even using the hybrid-selection scheme in the E-MWO does not guarantee that a small value of fitness (SSE) is achieved, but it still produces a very-well trained ANN that is able to gain the best (or an acceptable) classification accuracy. It can be noticed from the resutls, achieving less SSE does not always guarantee gaining best classification accuracy.

7.2 Comparison of the E-MWO with the Rival Training Methods

The performance of the E-MWO is compared against the original MWO and the other rival training algorithms, HS-BtW, GA, PSO, and BP. All experiments are conducted using the same configurations explained in Section 6. The results show that the classification accuracy of the ANN achieved by the E-MWO training method is either better or close to the other training methods. However, the training time is considerably less in almost all classification problems. The E-MWO scores the best in terms of classification accuracy in three problems: Haberman, Magic, and Diabetes. In addition, the E-MWO scores the second rank (and very close to the first rank) for other two problems: Iris and Glass. It is known that the difference between the achieved accuracy by the E-MWO and the other training methods is significant, based on the *P* value of the *t*-test for these problems (i.e. Haberman, Magic, Diabetes).

The E-MWO fails to be in the first two ranks in the Cancer and Ionosphere problems; however, the best-accuracy out of 20 runs is higher than all the other training methods. Also, the *P* value of the *t*-test is larger

Table 4: The *P* Value of the *t*-test of Each Pair of E-MWO and the Other Training Methods for the Classification Accuracy.

Problem	(E-MWO, MWO)	(E-MWO, HS-BtW)	(E-MWO, GA)	(E-MWO, PSO)	(E-MWO, BP)	First rank	Second rank
Haberman	0.135	**1.974E–5**	**4.017E–6**	**2.448E–9**	**1.211E–9**	**E-MWO**	MWO
Iris	**1.926E–6**	**0.0156**	**1.714E–4**	**3.042E–3**	0.119×	BP	**E-MWO**
Magic	**8.954E–5**	**8.109E–4**	**5.747E–4**	0.422	0.2168	**E-MWO**	PSO
Diabetes	**4.702E–6**	**5.576E–6**	**1.856E–7**	**2.148E–6**	**1.508E–5**	**E-MWO**	BP
Cancer	0.329	3.333E–6	0.047	0.211	**0.0197**	HS-BtW	GA
Ionosphere	**1.05E–6**	0.259×	**8.6E–4**	0.068	8.811E–4	BP	HS-BtW
Glass	**2.1E–8**	**9.161E–7**	**1.850E–10**	0.059×	**1.960E–4**	PSO	**E-MWO**
Thyroid	**0.040**	**0.035**	0.063	1.814E–4	1.269E–30	BP	PSO

The first rank and second rank columns denote the method that ranked first and second based on the calculated mean values of accuracies.

Table 5: The *P* Value of the *t*-test of Each Pair of E-MWO and the Other Training Methods for the Training Time.

Problem	(E-MWO, MWO)	(E-MWO, HS-BtW)	(E-MWO, GA)	(E-MWO, PSO)	(E-MWO, BP)	First rank	Second rank
Haberman	**2.32E–21**	**0.01111**	**0.0282**	**1.400E–13**	**2.234E–7**	**E-MWO**	GA
Iris	**3.74E–12**	**0.00439**	**0.0013**	**7.436E–16**	**3.856E–36**	**E-MWO**	GA
Magic	0.398×	0.46370	**5.964E–8**	**1.473E–7**	**5.941E–17**	MWO	**E-MWO**
Diabetes	**2.89E–12**	**5.334E–6**	**1.566E–6**	**2.791E–7**	**4.722E–14**	**E-MWO**	MWO
Cancer	**8.25E–12**	**1.422E–8**	**1.611E–36**	**6.977E–14**	**8.643E–16**	**E-MWO**	MWO
Ionosphere	**3.58E–11**	**5.35E–12**	**5.814E–19**	**5.021E–7**	**3.736E–23**	**E-MWO**	MWO
Glass	**2.762E–8**	**9.01E–12**	**3.489E–15**	**3.865E–13**	**1.32E–4**	**E-MWO**	MWO
Thyroid	0.2161	8.602E–6	0.4344	**2.604E–18**	**1.170E–29**	HS-BtW	**E-MWO**

The first rank and second rank columns denote the method that ranked first and second based on the calculated mean values of training time.

Table 6: Benchmark Classification Problems' Results for 20 run.

Algorithm		Iter.	SSE	Tr. time	Acc.%	Algorithm		Iter.	SSE	Tr. time	Acc.%
Haberman classification problem						**Iris classification problem**					
E-MWO	Best	156	524.0	**4.0**	**83.87**	E-MWO	Best	380.0	232.0	**7.0**	**100.0**
	Mean	148.5	502.2	**4.05**	**78.06**		Mean	176.7	207.0	**3.3**	95.5
MWO	Best	1000	356.9	16.0	79.03	MWO	Best	1000	41.3	12.0	96.67
	Mean	1000	352.4	16.2	77.1		Mean	1000	83.4	14.5	87.83
HS-BtW	Best	1181	400.8	8.0	75.81	HS-BtW	Best	20,000	9.5	91.0	100.0
	Mean	4169	384.1	27.35	69.354		Mean	3417	31.4	15.7	90.99
GA	Best	200	456.0	10.0	77.42	GA	Best	295	48.0	10.0	96.67
	Mean	123	511.2	6.5	74.11		Mean	235	87.0	8.0	84.66
PSO	Best	1500	420.0	29.0	77.42	PSO	Best	1500	24.0	38.0	96.67
	Mean	1500	303.8	17.1	71.21		Mean	1500	8.1	26.0	92.17
BP	Best	1001	321.0	1285.0	74.19	BP	Best	1225	7.8	1132.0	96.67
	Mean	1225	304.9	1621.7	71.85		Mean	1037	7.8	826.4	**96.67**
Magic classification problem						**Diabetes classification problem**					
E-MWO	Best	249.0	22,476.0	263.0	**93.01**	E-MWO	Best	84.0	1484.0	**10.0**	**89.61**
	Mean	290.5	16,934.6	340.5	**83.68**		Mean	219.2	1481.0	**28.75**	**81.01**
MWO	Best	1000	8929.5	**193.0**	80.18	MWO	Best	1000	806.8	69.0	78.57
	Mean	1000	9968.9	**202.45**	77.99		Mean	1000	832.0	69.0	75.1
HS-BtW	Best	4272	10,448.3	295.0	80.86	HS-BtW	Best	2489	890.6	25.0	78.57
	Mean	4693	10,807.2	350.7	79.19		Mean	11,308	842.1	113.9	75.25
GA	Best	2392	11,200.0	2847.0	80.81	GA	Best	1317	992.0	175.0	79.22
	Mean	2445	12,179.0	1997.3	79.02		Mean	2961	1025.4	392.3	73.89
PSO	Best	1500	6757.3	502.0	85.65	PSO	Best	1500	647.4	81.0	76.62
	Mean	1500	7622.0	761.8	83.41		Mean	1500	762.4	124.5	75.03
BP	Best	842	6137.3	42,542.0	84.02	BP	Best	4962	407.9	16,311.0	79.22
	Mean	756	6264.0	31,815.0	82.60		Mean	4200	482.4	14,112.2	75.42
Cancer classification problem						**Ionosphere classification problem**					
E-MWO	Best	150.0	412.0	**19.0**	98.54	E-MWO	Best	184.0	668.0	**8.0**	**95.77**
	Mean	169.8	307.2	**22.15**	96.86		Mean	216.8	505.8	**10.45**	89.65
MWO	Best	1000	146.5	44.0	98.54	MWO	Best	1000	248.6	22.0	92.96
	Mean	1000	134.1	45.35	96.72		Mean	1000	283.9	22.75	83.8
HS-BtW	Best	8355	97.5	125.0	**99.29**	HS-BtW	Best	6842	77.6	69.0	94.37
	Mean	8804	98.2	129.6	**98.42**		Mean	14,505	76.1	117.3	90.28
GA	Best	1000	128	1355.0	99.29	GA	Best	2880	224.0	281.0	92.96
	Mean	9807	176.5	1329.4	97.46		Mean	9005	228.4	878.5	86.55
PSO	Best	1500	88.0	135.0	97.81	PSO	Best	1500	300.0	71.0	94.37
	Mean	1500	67.9	148.9	96.61		Mean	1500	149.6	61.9	87.75
BP	Best	1103	24.61	3909.0	97.86	BP	Best	699	8.4	1295.0	**95.77**
	Mean	1348	23.42	4097.3	96.17		Mean	913	13.8	1472.0	**92.04**
Glass classification problem						**Thyroid classification problem**					
E-MWO	Best	585.0	744.0	**50.0**	85.71	E-MWO	Best	215.0	4644.0	563.0	93.13
	Mean	271.9	838.2	**20.9**	70.6		Mean	354.7	4336.0	1061.2	92.65
MWO	Best	1000	386.8	45.0	61.9	MWO	Best	1000	3237.1	1236.0	92.57
	Mean	1000	447.7	42.9	50.36		Mean	1000	3257.7	1223.4	92.55
HS-BtW	Best	20,000	337.8	134.0	72.09	HS-BtW	Best	480	3225.6	**78.0**	92.64
	Mean	17,369	445.2	114.9	52.44		Mean	413	3244.6	**67.4**	92.55
GA	Best	10,000	543.7	740.0	62.79	GA	Best	268	3416.0	828.0	92.57
	Mean	8650	595.5	639.7	45.23		Mean	353	3416.0	1103.2	92.57
PSO	Best	1500	592.0	143.0	**97.62**	PSO	Best	1500	1972.0	5064.0	**99.79**
	Mean	1500	484.8	114.4	**77.38**		Mean	1500	2782.2	4766.8	94.77
BP	Best	6505	62.4	6104.0	72.09	BP	Best	1121	613.7	36,297.0	97.29
	Mean	3187	153.7	3003.1	60.11		Mean	980	687.9	32,008.6	**96.27**

Iter. : Number of iterations, Tr. time: training time in seconds, Acc.: classification accuracy.

than α, which means that even the E-MWO achieves a lower accuracy, but the difference is not significant from the other training methods.

The merits of the E-MWO is spread over all types of the benchmark problems; it shows excellent accuracy in small- and medium-size problems (e.g. Haberman, Iris, Magic, and Diabetes) and acceptable accuracy in large-size problems (e.g. Glass). Also, the performance of the E-MWO is good in the unbalanced-class problems (e.g. Haberman and Diabetes), multi-class problems (e.g. Glass and Iris), and problems that have a large number of patterns (e.g. Magic). Table 4 presents the *P* value of the *t*-test that were performed on the E-MWO and the rival training methods for the classification accuracy. The bold values indicate that the mean value of the E-MWO is superior and significant, while the values with a cross sign ($^{\times}$) indicate that the mean value of the E-MWO is slightly lower than the other method, but the difference is not significant.

On the other hand, the E-MWO achieves the lowest mean training time in six out of eight problems due to the property of fast convergence inherited from the original MWO and also the added or modified parts over the MWO such as using the dynamic termination criterion. The E-MWO terminates its process when it notices that the candidate solutions could not improved further. The E-MWO fails to obtain the least training time for problems that have a large number of patterns (i.e. Magic and Thyroid); however, it ranked second in these two problems. Table 5 presents the *P* value of the *t*-test for the training time, while the detailed results are given in Table 6.

In a comparing the E-MWO with the original MWO, the E-MWO scores the best in all eight benchmarking problems in terms of accuracy. This enhancement in accuracy is due to the proposed solutions in the E-MWO for the shortcomings of the MWO. The training time of the E-MWO is lower than the MWO for seven problems out of eight. This enhancement in training time is a result of using the dynamic termination criterion, which incurs less number of iterations. In addition, the enhancement of time is a result of performing the update position and fitness calculation for the mussels that have a moving probability $P_i(t) = 1$ only, instead of performing these calculations for all mussels regardless of whatever the value of the mussel moving probability is.

8 Conclusion

In this paper, the enhanced version of the MWO (the E-MWO algorithm) is demonstrated, analyzed, and discussed. The E-MWO adaptively and dynamically sets the value of the shape parameter μ. It has a multi-step length approach and a new hybrid-selection scheme to update the mussel population positions. The termination criterion depends on two new dynamic quality measures: S_R and U_R.

The feed-forward ANN has been trained by adapting the MWO and the E-MWO algorithms. The pattern classification problem of the different datasets has been tackled in this research to validate and test the algorithm efficiency. Two criteria are considered during the evaluation process: classification accuracy and training time.

The results indicated that the E-MWO algorithm scores the best against the other rival training algorithms in terms of classification accuracy in three out of eight problems significantly and at par with the other methods for the remaining problems. The E-MWO scores the best in terms of training time in six out of eight problems significantly and very close to other methods for the other two problems.

The increment made to the number of algorithm parameters is considered as a shortcoming for the E-MWO. This shortcoming could be considered as a future work for this paper. In addition, it would be valuable to extend this research by considering other applications of the ANN such as prediction, clustering, and medical image diagnosis. Some of the training methods that exist in the literature represent a hybrid between the P-Metaheuristic and GD, such as the GA with BP [5] and PSO with BP [51]. It is interesting to hybridize the E-MWO with GD algorithms or with local search algorithms such as simulated annealing to increase the fine-tuning capabilities.

Acknowledgment: The authors would like to thank Dr. Moh'd Khaled Shambour and Mr. Basem O. Alijla for the valuable comments they provided. This research is supported by UNIVERSITI SAINS MALAYSIA and has been funded by the Research University Cluster (RUC) grant titled by "Reconstruction of the Neural Microcircuitry of Reward-Controlled Learning in the Rat Hippocampus" (1001/PSKOM/8630022).

Appendix A: Determination of the *Best-M* Subset

Assume that the steady state is detected, the mussel population is $N=8$, and the random mussel is selected from the first $M=50\%$ of the mussels. The *best-M* subset size is $N\times M=8\times 0.5=4$ mussels. Note that the global-best mussel is $m_g \notin$ *best-M* if the steady state is detected as illustrated in Table 7.

Appendix B: Calculations of the iS_R and the S_R

Table 8 illustrates how the iS_R and the S_R are calculated for a sample of three iterations, assuming that the mussel population size is $N=5$.

Table 7: Example for Determination of the *best-M* Subset.

Mussel rank	Fitness (m_i)	$m_i \in$ *best-M*
1	4.6	No
2	5.6	Yes
3	7.1	Yes
4	10.2	Yes
5	11.9	Yes
6	13.6	No
7	15.1	No
8	27.2	No

Table 8: Sample Calculations of the iS_R and the S_R.

Iteration (t)	Mussel no.	Fitness (m_i)	iS_R	S_R
1	1	4.6	$\frac{2}{5}=0.4$	$\frac{0.4}{1}=0.4$
	2	4.6		
	3	5.1		
	4	7.2		
	5	11.9		
2	1	3.2	$\frac{0}{5}=0.0$	$\frac{0.4+0.0}{2}=0.2$
	2	5.8		
	3	7.4		
	4	9.8		
	5	11.1		
3	1	2.4	$\frac{4}{5}=0.8$	$\frac{0.4+0.0+0.8}{3}=0.4$
	2	2.4		
	3	3.8		
	4	5.1		
	5	5.1		

Bibliography

[1] A. A. Abusnaina and R. Abdullah, Mussels wandering optimization algorithm based training of artificial neural networks for pattern classification, in: *Proceedings of the 4th International Conference on Computing and Informatics*, pp. 78–85, Malaysia, 2013.

[2] A. A. Abusnaina, R. Abdullah and A. Kattan, Enhanced MWO training algorithm to improve classification accuracy of artificial neural networks, in: *Recent Advances on Soft Computing and Data Mining*, pp. 183–194, Springer International Publishing, Cham, Malaysia, 2014.

[3] A. A. Abusnaina, R. Abdullah and A. Kattan, The application of mussels wandering optimization algorithm for spiking neural networks training, in: *International Engineering Conference (IEC2014) on Developments in Civil and Computer Engineering Applications*, pp. 197–204, Iraq, 2014.

[4] S. Agatonovic-Kustrin and R. Beresford, Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research, *Pharm. Biomed. Anal.* **22** (2000), 717–727.

[5] E. Alba and J. F. Chicano, Training neural networks with GA hybrid algorithms, in: *Genetic and Evolutionary Computation GECCO*, pp. 852–863, Springer, Berlin Heidelberg, 2004.

[6] J. An, Q. Kang, L. Wang and Q. Wu, Mussels wandering optimization: an ecologically inspired algorithm for global optimization, *Cognit. Comput.* **5** (2013), 188–199.

[7] J. An, S. Liu, Q. Kang and W. Yan, Time-sharing characteristic clustering analysis of household energy consumption via K-mussels wandering optimization, *Sens. Lett.* **12** (2014), 270–274.

[8] K. Bache and M. Lichman, UCI Machine Learning Repository [online], (accessed on February, 2013). University of California, Irvine, School of Information and Computer Sciences, Available: http://archive.ics.uci.edu/ml.

[9] C. Bennett, R. A. Stewart and C. D. Beal, ANN-based residential water end-use demand forecasting model, *Expert Syst. Appl.* **40** (2013), 1014–1023.

[10] C. Blum and K. Socha, Training feed-forward neural networks with ant colony optimization: an application to pattern classification, in: *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, IEEE, Brazil, 2005.

[11] J. A. Bullinaria and K. AlYahya, Artificial Bee Colony training of neural networks, in: *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, pp. 191–201, Springer International Publishing, Cham, UK, 2014.

[12] E. Cantu-Paz and C. Kamath, An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems, *IEEE Trans. Syst. Man Cybern. B Cybern.* **35** (2005), 915–927.

[13] T. G. Crainic and M. Toulouse, Parallel meta-heuristics, in: *Handbook of Metaheuristics*, pp. 497–541, Springer, US, 2010.

[14] M. De Jager, F. Bartumeus, A. Klzsch, F. J. Weissing, G. M. Hengeveld, B. A. Nolet and J. van de Koppel, How superdiffusion gets arrested: ecological encounters explain shift from Lvy to Brownian movement, *Proc. R. Soc. B Biol. Sci.* **281** (2014), 2013–2605.

[15] V. K. Dhar, A. K. Tickoo, R. Koul and B. P. Dubey, Comparative performance of some popular artificial neural network algorithms on benchmark and function approximation problems, *PRAMANA J. Phys. Ind. Acad. Sci.* **74** (2010), 307–324.

[16] R. E. Dorsey, J. D. Johnson and W. J. Mayer, A genetic algorithm for the training of feedforward neural networks, *Adv. Artif. Intell. Econ. Finance Manag.* **1** (1994), 93–111.

[17] G. I. Evers and M. Ben Ghalia, Regrouping particle swarm optimization: a new global optimization algorithm with improved performance consistency across benchmarks, in: *IEEE International Conference on Systems, Man and Cybernetics, SMC 2009*, pp. 3901–3908, IEEE, USA, 2009.

[18] Q. Gao, K. Q. Y. Lei and Z. He, An improved genetic algorithm and its application in artificial neural network, in: *Fifth International Conference on Information, Communications and Signal Processing*, pp. 357–360, IEEE, Thailand, 2005.

[19] M. Gilli and P. Winker, *A review of heuristic optimization methods in econometrics*, Swiss Finance Institute Research Paper, pp. 8–12, Switzerland, 2008. Available at SSRN: http://ssrn.com/abstract=1140655.

[20] R. C. Green II, L. Wang and M. Alam, Training neural networks using central force optimization and particle swarm optimization: insights and comparisons, *Expert Syst. Appl.* **39** (2012), 555–563.

[21] D. Gupta and S. Ghafir, An overview of methods maintaining diversity in genetic algorithms, *Int. J. Emerg. Technol. Adv. Eng.* **2** (2012), 56–60.

[22] S. He, Q. H. Wu and J. R. Saunders, Group search optimizer: an optimization algorithm inspired by animal searching behavior, *IEEE Trans. Evol. Comput.* **13** (2009), 973–990.

[23] S. He, Q. H. Wu and J. R. Saunders, Breast cancer diagnosis using an artificial neural network trained by group search optimizer, *Trans. Inst. Meas. Control* **31** (2009), 517–531.

[24] G. B. Huang, Q. Y. Zhu and C. K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: *Proceedings of IEEE International Conference on Neural Networks 2*, pp. 985–990, IEEE, Hungary, 2004.

[25] S. Ibric, M. Jovanovi, Z. Djuri, J. Paroji, S. D. Petrovi, L. Solomun and B. Stupar, Artificial neural networks in the modeling and optimization of aspirin extended release tablets with Eudragit L 100 as matrix substance, *AAPS PharmSciTech.* **4** (2003), 62–70.

[26] T. Ince, S. Kiranyaz, J. Pulkkinen and M. Gabbouj, Evaluation of global and local training techniques over feed-forward neural network architecture spaces for computer-aided medical diagnosis, *Expert Syst. Appl.* **37** (2010), 8450–8461.

[27] T. Jayalakshmi and A. Santhakumaran, Statistical normalization and back propagation for classification, *Int. J. Comput. Theory Eng.* **3** (2011), 1793–8201.
[28] D. Karaboga, B. Akay and C. Ozturk, Articial bee colony (ABC) optimization algorithm for training feed-forward neural networks, in: *4th International Conference on Modeling Decisions for Artificial Intelligence MDAI*, pp. 318–329, Springer-Verlag, Berlin, Heidelberg, 2007.
[29] N. K. Kasabov, *Foundations of neural networks, fuzzy systems, and knowledge Engineering, Second printing, A Bradford Book*, The MIT Press, Cambridge, MA, London, England, 1998.
[30] A. Kattan and R. Abdullah, Training of feed-forward neural networks for pattern-classification applications using music inspired algorithm, *Int. J. Comput. Sci. Inf. Secur.* **9** (2011), Malaysia, 44–57.
[31] A. Kattan and R. Abdullah, Training feed-forward artificial neural networks For pattern-classification using the harmony search algorithm, in: *The Second International Conference on Digital Enterprise and Information Systems*, pp. 84–97, Malaysia, 2013.
[32] A. Kattan, R. Abdullah and R. A. Salam, Harmony search based supervised training of artificial neural networks, in: *IEEE International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pp. 105–110, IEEE, UK, 2010.
[33] S. Kiranyaz, T. Ince, A. Yildirim and M. Gabbouj, Evolutionary artificial neural networks by multi-dimensional particle swarm optimization, *Neural Netw.* **22** (2009), 1448–1462.
[34] S. Kulluk, L. Ozbakir and A. Baykasoglu, Self-adaptive global best harmony search algorithm for training neural networks, *Procedia Comput. Sci.* **3** (2011), 282–286.
[35] S. Kulluk, L. Ozbakir and A. Baykasoglu, Training neural networks with harmony search algorithms for classification problems, *Eng. Appl. Artif. Intell.* **25** (2012), 11–19.
[36] F. Liang, Annealing stochastic approximation Monte Carlo algorithm for neural network training, *Mach. Learn.* **68** (2007), 201–233.
[37] C. S. Lin, Toward a new three layer neural network with dynamical optimal training performance. in: *Proceedings IEEE International Conference on Systems, Man and Cybernetics*, pp. 3101–3106, Montreal, Quebec, Canada, 2007.
[38] Y. Liu and X. Yao, A population-based learning algorithm which learns both architectures and weights of neural networks, *Chin. J. Adv. Softw. Res.* **3** (1996), 54–65.
[39] D. Manjarres, I. Landa-Torres, S. Gil-Lopez, J. Del Ser, M. N. Bilbao, S. Salcedo-Sanz and Z. W. Geem, A survey on applications of the harmony search algorithm, *Eng. Appl. Artif. Intell.* **26** (2013), 1818–1831.
[40] M. Mavrovouniotis and S. Yang, Training neural networks with ant colony optimization algorithms for pattern classification, *Soft Comput.* **19** (2014), 1–12.
[41] R. Mendes, P. Cortez, M. Rocha and J. Neves, Particle swarms for feedforward neural network training, in: *Proceedings of the IEEE International Joint Conference on Neural Networks, IJCNN'02*, pp. 1895–1899, IEEE, USA, 2002.
[42] D. J. Montana and L. Davis, Training feedforward neural networks using genetic algorithms, *IJCAI* **89** (1989), 762–767.
[43] P. C. Pendharkar and J. A. Rodger, An empirical study of impact of crossover operators on the performance of non-binary genetic algorithm based neural approaches for classification, *Comput. Oper. Res.* **31** (2004), 481–498.
[44] W. Schiffmann, M. Joost and R. Werner, Application of genetic algorithms to the construction of topologies for multilayer perceptrons, in: *Artificial Neural Nets and Genetic Algorithms*, pp. 675–682, Springer, Vienna, 1993.
[45] D. Silva, L. Pacifico and T. Ludermir, An evolutionary extreme learning machine based on group search optimization, in: *IEEE Congress of Evolutionary Computation*, pp. 574–580, USA, 2011.
[46] K. Socha and C. Blum, An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training, *Neural Comput. Appl.* **16** (2007), 235–247.
[47] Z. Song, B. Murray, B. Sammakia and S. Lu, Multi-objective optimization of temperature distributions using artificial neural networks, in: *13th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pp. 1209–1218, USA, 2012.
[48] B. Trawinski, M. Smtek, Z. Telec and T. Lasota, Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms, *Int. J. Appl. Math. Comput. Sci.* **22** (2012), 867–881.
[49] A. B. Van Wyk and A. P. Engelbrecht, Overfitting by PSO trained feedforward neural networks, in: *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, Spain, 2010.
[50] G. Wei, Study on evolutionary neural network based on ant colony optimization, in: *International Conference on International Conference on Computational Intelligence and Security Workshops*, pp. 3–6, China, 2007.
[51] M. Yaghini, M. M. Khoshraftar and M. Fallahi, A hybrid algorithm for artificial neural network training, *Eng. Appl. Artif. Intell.* **26** (2013), 293–301.
[52] P. Yan, S. Liu, Q. Kang, B. Huang and M. Zhou, A data clustering algorithm based on mussels wandering optimization, in: *IEEE 11th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 713–718, USA, 2014.
[53] J. Yu, S. Wang and L. Xi, Evolving artificial neural networks using an improved PSO and DPSO, *Neurocomputing* **71** (2008), 1054–1060.
[54] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, in: *Mendel 9th International Conference Soft Computing*, pp. 41–46, Czech Republic, 2003.