CLASSIFICATION OF SONAR DATA SET USING NEURAL NETWORK TRAINED BY GRAY WOLF OPTIMIZATION

M.R. Mosavi, M. Khishe, A. Ghamgosar*

Abstract: Multi-Layer Perceptron Neural Networks (MLP NNs) are the commonly used NNs for target classification. They purposes not only in simulated environments, but also in actual situations. Training such NNs has significant importance in a way that many researchers have been attracted to this field recently. Conventional gradient descent and recursive method has long been used to train NNs. Improper classification accuracy, slow convergence speed and trapping in local minimums are disadvantages of the traditional methods. In order to overcome these issues, in recent years heuristic and meta-heuristic algorithms are widely used. This paper uses Gray Wolf Optimization (GWO) algorithm for training the NN. This algorithm is inspired by lifestyle and hunting method of GWs. GWO has a superior ability to solve the high-dimension problems, so we try to classify the Sonar dataset using this algorithm. To test the proposed method, this algorithm is compared to Particle Swarm Optimization (PSO) algorithm, Gravitational Search Algorithm (GSA) and the hybrid algorithm (i.e. PSOGSA) using three sets of data. Measured metrics are convergence speed, the possibility of trapping in local minimum and classification accuracy. The results show that the proposed algorithm in most cases provides better or comparable performance compared to the other mentioned algorithms.

Key words: *classification, sonar, Multi-Layer Perceptron Neural Network, Grey Wolf Optimization, Particle Swarm Optimization, Gravitational Search Algorithm*

Received: December 13, 2014
Revised and accepted: June 11, 2016

DOI: 10.14311/NNW.2016.26.023

1. Introduction

Multi-Layer Perceptron Neural Networks (MLP NNs) are among the most versatile tools for soft computing. Non-linear problems can be solved using these networks. In general, NNs are used for pattern classification, data prediction and different

*Mohammad Reza Mosavi – Corresponding author, Mohammad Khishe, Abolfazl Ghamgosar, Department of Electrical Engineering, Iran University of Science and Technology, Narmak, Tehran 16846-13114, Iran, E-mail: m_mosavi@iust.ac.ir, m_khishe@elec.iust.ac.ir, Ghamgosar@elec.iust.ac.ir

functions approximation [1, 2, 29, 33]. Pattern classification is defined as classifying the data into the predefined discrete sets [3], whereas prediction is defined as predicting the future events based on present and past data [10]. Finally, process of modeling the relationships between input variables is included in function approximation. It is proved that MLP NNs with one hidden layer have the ability of approximating any continuous or discrete function [4]. Regardless of the applications, learning is the distinctive ability of them [33]. This means that these networks are capable of learning from an experience or experiment which is similar to human brain. This feature (learning) is an essential part of all NNs which may be divided into two types: supervised learning [8] and unsupervised learning [34].

For training the MLP NNs (in most cases), optimized Back-Propagation (BP) [45] or standard algorithms [14] which are categorized under supervised learning methods are used. Gradient based BP algorithm has some drawbacks such as: (a) slow convergence rate, (b) high probability of local optima entrapment, (c) highly depends on the initial learning rate and the size of each step as inappropriate values of these variables may results in algorithm divergence [32], and (d) limited range usage [21].

Therefore, it is not reliable for practical applications. This problem has been addressed by many researchers [11], but no remarkable result regarding the optimization has been achieved and each method has only its side effects. Prior researches show that the heuristic and meta-heuristic search algorithms can be suitable substitutes for gradient based methods, since the stochastic nature of these methods reduces the error percentage and possibility of trapping in local minimum compared to gradient based methods. In general, meta-heuristic methods can be divided into two main categories: single-solution and multi-solution [42]. In single-solution category, the search process begins with a candidate solution which improves in the next iterations. An example of the single-solution-based optimizers is Simulated Annealing (SA) [18]. On the other hand, in multi-solution category, optimization is performed using a set of solutions (population). In this case, the search process begins with a random initial population (multiple solutions) and improves as the algorithm iterates. Some of the most popular multi-solution trainers in the literature are: Genetic Algorithm (GA) [43], Ant Colony Optimization (ACO) [6], Particle Swarm Optimization (PSO) [17], Artificial Bee Colony (ABC) [16], Differential Evolution (DE) [15, 39], Hybrid Gravitational Search Algorithm and Particle Swarm Optimization (GSA-PSO) [24, 25], Biogeography Based Optimization algorithm (BBO) [29], Stochastic Fractal Search (SFS) [38] Charged System Search (CSS) [35], Ant Lion Optimizer (ALO) [23] and Multi-Verse Optimization (MVO) trainer [27]. Multi-solution meta-heuristic methods have several advantages compared to single-solution algorithms:

(I) Multi-solution algorithms share their information about search space, which leads to sudden mutations towards the promising and satisfactory part of the search space.

(II) Multi-solution algorithms have the ability to avoid local optimal solutions.

(III) Exploring and identifying capabilities of multi-solution meta-heuristic approaches are higher than single-solution algorithms.

We can divide all existing algorithms into three major categories: Swarm Intelligence (SI) based, bio-inspired (but not SI-based) and physics/chemistry-based. The main defect of the first two groups, is their high computational complexity. For this reason, we focus on third group (SI). SI is inspired by natural colonies, herds and groups. Some of the most popular techniques of SI are ACO [43], PSO [6] and ABC [17]. Some of the advantages of SI algorithms are as follows:

(I) SI algorithms maintain the information regarding search space during each iteration period, whereas the Evolutionary Algorithms (EA) does not utilize the information about previous generations.

(II) SI algorithms often use the memory to store the best solution obtained.

(III) SI algorithms have typically less parameters to be adjusted.

(IV) SI algorithms have fewer operators with respect to evolutionary methods (reproduction, mutation, and etc.).

(V) Implementation of SI algorithms is rather easy.

Regardless of the differences between the various meta-heuristic methods, they share a common feature of dividing the search process into identification and utilization stages [19, 23, 27, 28]. Identification phase refers to the extensive review of desired and promising results of search space. Any algorithm requires stochastic operators to be able to search randomly in overall search space and support the identification phase. Utilization phase refers to local searching capability in the areas which are obtained from identification phase. Finding an appropriate balance between two phases is a challenging problem due to the random nature of meta-heuristic methods.

Physics-based algorithms like GSA have a good performance in exploration phase and search the entire search space widely and have a slow performance in exploitation phase because objects become heavy in this phase. SI based algorithms do not have a good performance at the prime phases because of lack of SI (weak in exploration phase), however their performance improves gradually as they collect SI that means they have a good performance in exploitation phase. PSOGSA algorithm combines the ability of PSO (in exploitation phase) and GSA (in exploration phase) algorithms and hereby its performance improves, however there is not a good balance between these two phases [19].

Reference [28] in detail showed that the GWO outperforms other SI algorithms in balancing two mentioned phases, so it has a promising operation in finding global minima. On the other hand, GWO is very suitable for implementation. GWO algorithm mimics the leadership hierarchy and hunting mechanism of gray wolves in nature proposed by Mirjalili et al. in 2014 [28].

Because of unique capabilities of GWO algorithm, it has been used in many fields in the last two years such as: feature subset selection [7], DC motors control [20], economic emission dispatch problems [40], image registration [36], Radial Basis Function (RBF) networks training [30] and solving optimal reactive power dispatch problem [41]. According to the mentioned factors, we will also use the GWO algorithm to train the MLP network.

The rest of this paper is organized as follows. Section 2 introduces the MLP NNs. Section 3 discusses the general aspects of Gray Wolf Optimization (GWO) technique. Application of GWO algorithm as a meta-heuristic learning algorithm in MLP NNs has been described in Section 4. The results are illustrated in Section 5. Finally, conclusion and corresponding research topics that can be pursued are presented in Section 6.

2. Multi-Layer Neural Network

Fig. 1 shows a MLP NN with three layers. n is the number of input nodes, h is the hidden nodes and m is the number of output nodes. As can be seen, there exist one-way connection between nodes in a MLP NN which is categorized under the feed-forward NNs family. The MLP NN outputs are calculated using the Eq. (1):

$$p_j = \sum_{i=1}^{n} (w_{ij} x_i) - \theta_j, \qquad j = 1, 2, \ldots, h, \tag{1}$$

where w_{ij} is the connection weight from the i-th node (in input layer) to the j-th node (in hidden layer), θ_j is the j-th node's bias (in hidden layer) and x_i is the i-th node's input (in input layer). Each hidden layer node's output is calculated using a tanh function as given in Eq. (2):

$$s_j = \tanh(p_j) \qquad j = 1, 2, \ldots, h. \tag{2}$$

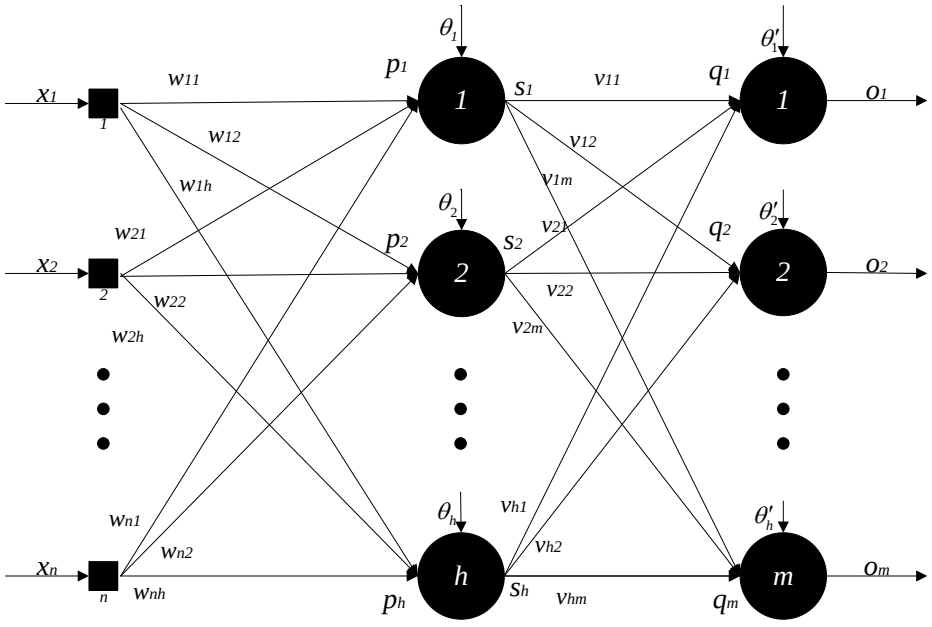

Fig. 1 *A MLP NN with one hidden layer.*

Inputs and outputs of the output layer can be defined as Eqs. (3) and (4), respectively:

$$q_k = \sum_{j=1}^{h} (v_{jk} s_j) - \theta'_k, \qquad k = 1, 2, \ldots, m, \tag{3}$$

$$o_k = \tanh(q_k) \qquad k = 1, 2, \ldots, m, \tag{4}$$

where v_{jk} is the connection weight from the j-th node (in hidden layer) to k-th node (in output layer) and θ'_k is the k-th node's bias (in output layer). The most important parts of the MLP NNs are the connection weights and node's biases. As one can see from above equations, the final output of the network is defined by edge weights and node's biases. Training the MLP NN includes finding the best values for connection weights and node's biases, so that the specific inputs result in desired outputs.

3. Gray Wolf Optimizer

In this section, inspiration source of the presented method has been introduced and its mathematical model has been provided.

3.1 Inspiration source

GW belongs to the family of canines. GWs are at the apex of predators, meaning that they are at the top of food chain. GWs would rather to live as a group. The average group size is between 5 and 12 [28]. They have a very hard-dominant social hierarchy which is shown in Fig. 2.

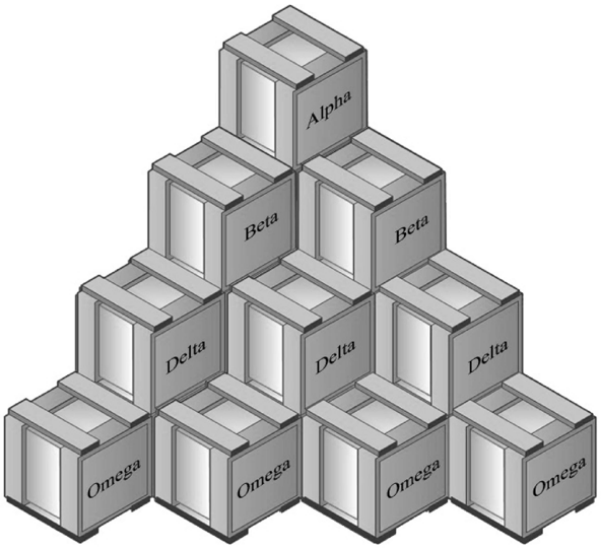

Fig. 2 *GW hierarchy (dominance decreases from top to bottom).*

Leaders group consists of one male and one female and is called alpha. Alpha is mostly responsible for making decisions about hunting, sleeping, walking time and so on. Alpha decisions are dictated to rest of group, yet somehow a democratic behavior is also observed that an alpha wolf follow the other wolves in the group. In group gatherings, the entire group will hold their tail downwards as a sign of recognizing the alpha. Given that alpha commands must be followed by other wolves, the alpha wolf is also known as the dominant wolf [22, 28]. Alpha wolves are the only ones allowed to select a mate within the group. It is interesting that an alpha is not necessarily the strongest member of the group, but it is best member from management point of view. This shows that the order of a group is much more important than its power.

Betas are in the second level of GWs hierarchy. Beta wolves are subordinates of the alpha group. They help alpha's in decision making or other activities related to group. Beta wolf can be a male or a female and if an alpha becomes too old or dies. It is the best candidate to replace alpha. Beta wolf should respect the alpha and rule the lower level wolves. In fact, beta is the counselor of the alpha as well as the group organizer. Beta executes the alpha commands throughout the group and refers the feedback to alpha.

The omega has the lowest rank in GWs hierarchy. Omega plays the role of victim in group. Omega wolf must always be obedient to the dominant wolves. They are the last group of wolves which are allowed to eat. It seems that omega does not play a significant role in the group, but it is observed that elimination of omega group has led to civil war and other some other problems. This is due to the depletion of violence and failure of other wolves through the omega's, which helps maintaining the structure of dominance and satisfaction of all group members.

If a wolf does not belong to alpha, beta or omega groups, it's called subordinate (or delta in some resources). Delta wolves should report to alpha's and beta's, but they have dominance over the omega. Scouts, guards, elders, hunters and caregivers belong to this category. Scouts oversee the boundaries of the territory and warn the group in case of any danger. Guardians are responsible to maintain and ensure the safety of the group. Elders are experienced wolves which were previously a member of alpha or beta groups. When hunting a prey and supplying the group food, hunters collaborate with alpha and beta groups. Finally, caregiver wolves are responsible for taking acre of poor, sick and wounded wolves. In addition to the social hierarchy, group hunting is another interesting social behavior of the GWs. According to Muro and his colleague's research [20], the main stages of hunting among GWs are as follows:

(I) Tracking, chasing and approaching the prey.

(II) Chasing, encircling and harassing the prey until it stops moving.

(III) Attacking the prey.

These stages are illustrated in Fig. 3. In this paper, GWs hunting method and their social hierarchy are mathematically modeled to design the GWO algorithm and perform the optimization process.

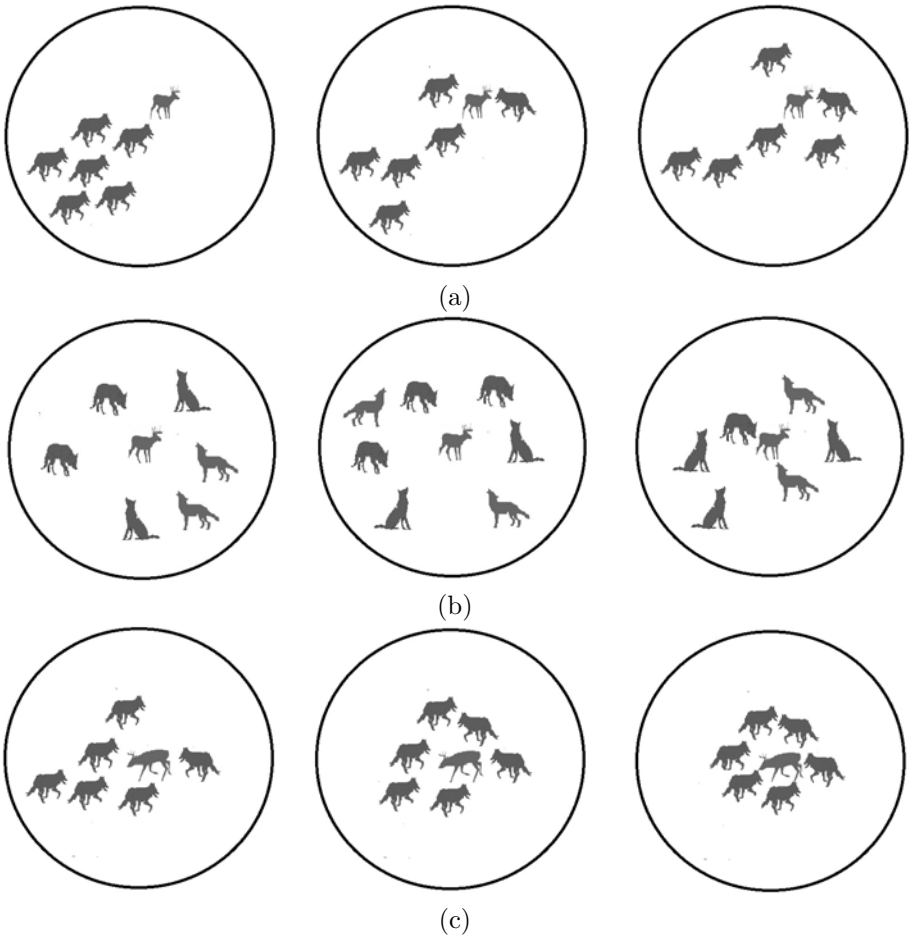

Fig. 3 *GWs hunting behavior: a tracking, chasing and approaching the prey, b chasing, surrounding and harassing the prey until it stops moving and c attacking the prey.*

3.2 Mathematical model and algorithm

In this section, mathematical models of social hierarchy, tracking, encircling and attacking are presented. Corresponding GWO algorithm is then specified [28].

3.2.1 Social hierarchy

When designing GWO algorithm, the most proper solution is assumed to be α, so that social hierarchy of the wolves can be mathematically model. The next best solutions are called beta (β) and delta (δ), respectively. The remaining candidate solutions are considered as omega (ω). The hunting process (optimization) in GWO algorithm is directed by α, β and δ. ω wolves follow these three groups.

3.2.2 Encircling the prey

As mentioned above, the prey is hunted during the siege. To mathematically model the encircling process, Eqs. (5) and (6) has been presented [28]:

$$\mathbf{d} = \left|\mathbf{c}\mathbf{x}_{\mathrm{p}}(t) - \mathbf{x}(t)\right|, \tag{5}$$

$$\mathbf{x}(t+1) = \mathbf{x}_{\mathrm{p}}(t) - \mathbf{a}\mathbf{d}, \tag{6}$$

where t represents the number of current iteration, $\mathbf{a}$ and $\mathbf{c}$ are the coefficient vectors, $\mathbf{x}_{\mathrm{p}}$ is the vector of prey position and $\mathbf{x}$ is the position vector of a GW. Both vectors $\mathbf{a}$ and $\mathbf{c}$ are calculated by the Eqs. (7) and (8), respectively:

$$\mathbf{a} = 2.\mathbf{f}.\mathbf{r}_1 - \mathbf{a}, \tag{7}$$

$$\mathbf{c} = 2.\mathbf{r}_2, \tag{8}$$

where $\mathbf{f}$ is reduced linearly from 2 to 0 through the iteration process. $\mathbf{r}_1$ and $\mathbf{r}_2$ are the random vectors in the range of [0,1]. In order to see the results, Eqs. (5) and (6) along with a two-dimensional vector and a number of possible neighbors are shown in Fig. 4a. As can be seen, a GW in position (x, y) can change its position with respect to prey's (x^*, y^*) location. Various locations around the most suitable agent can be obtained considering its current location and changing and setting the values of $\mathbf{a}$ and $\mathbf{c}$ vectors. For instance the location of $(x^* - x, y^*)$ is obtained by setting $\mathbf{a} = (1, 0)$ and $\mathbf{c} = (1, 1)$. Updated possible locations of a GW in a three-dimensional space are shown in Fig. 4b. It should be noted that the wolves are allowed to access any position between the points shown in Fig. 4 through the random vectors $\mathbf{r}_1$ and $\mathbf{r}_2$. So, any GW can randomly change its location within the space surrounding the prey using Eqs. (5) and (6).

This concept can be generalized to an n-dimensional search space. In this case movements of the GWs are done in a space which has more dimensions compared to previous section cube.

3.2.3 Hunting

The GW is capable of detecting the prey's location and encircling it. The hunting process is usually conducted by alpha wolves. Beta and delta wolves are occasionally participate in the hunting process. Unfortunately in an abstract search space there is no information about the optimum location (prey). In order to mathematically simulate the behavior of the GWs, it is assumed that the alpha (best solution available), beta and delta wolves are better informed about the location of potential prey. So, three of the best solutions yet obtained is stored and other agents are forced to update their positions according to the best agents locations. This relationship is expressed through the Eqs. (9), (10) and (11) [28]:

$$\mathbf{d}_\alpha = \left|\mathbf{c}_1\mathbf{x}_\alpha - \mathbf{x}\right|, \quad \mathbf{d}_\beta = \left|\mathbf{c}_2\mathbf{x}_\beta - \mathbf{x}\right|, \quad \mathbf{d}_\delta = \left|\mathbf{c}_3\mathbf{x}_\delta - \mathbf{x}\right|, \tag{9}$$

$$\mathbf{x}_1 = \mathbf{x}_\alpha - \mathbf{a}_1(\mathbf{d}_\alpha), \quad \mathbf{x}_2 = \mathbf{x}_\beta - \mathbf{a}_2(\mathbf{d}_\beta), \quad \mathbf{x}_3 = \mathbf{x}_\delta - \mathbf{a}_3(\mathbf{d}_\delta), \tag{10}$$

$$\mathbf{x}(t+1) = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3}{3}. \tag{11}$$

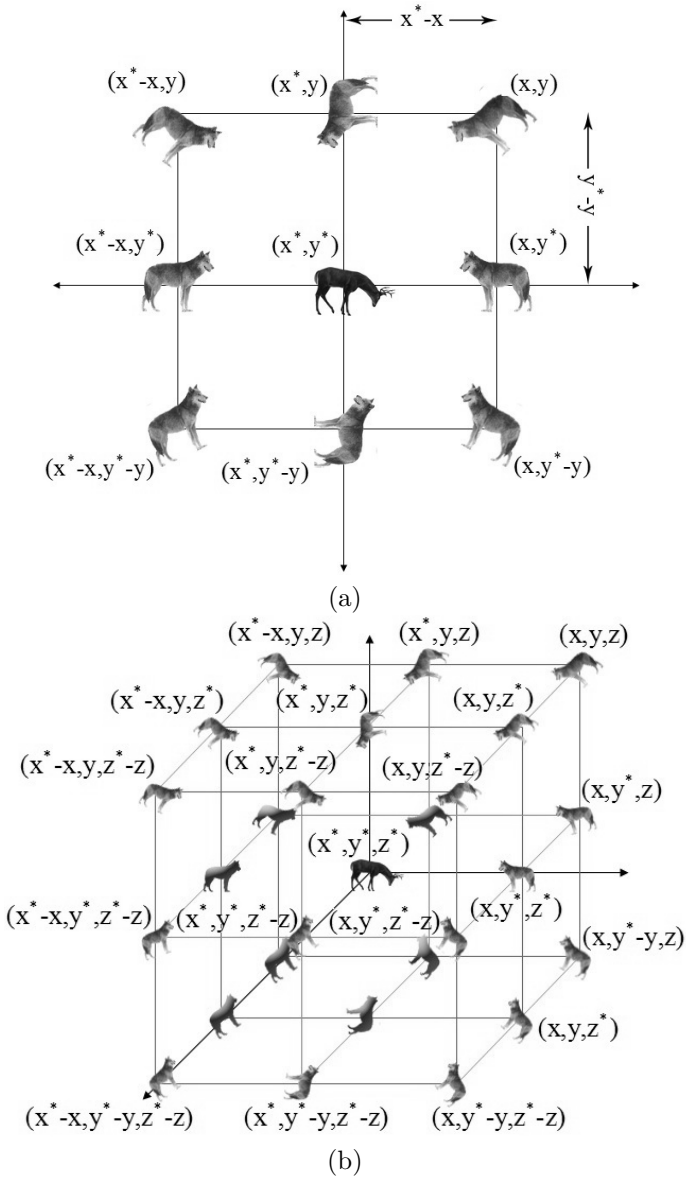

Fig. 4 *Two and three-dimensional vector space and their next possible positions [28].*

Fig. 5 shows the process of updating the search agent's location in two-dimensional search space regarding the position of alpha, beta and delta positions. As it can be seen, the final position is located randomly in a circle which is defined by alpha, beta and delta positions. In other words, the prey position is estimated by alpha, beta and delta groups and other wolves randomly update their positions within its vicinity.

3.2.4 Attacking the prey (utilization)

As mentioned above, the GWs will attack the pray and finish the hunt as soon as the prey stops moving. To mathematically model the approaching process, the value of **f** should be reduced. Note that the variation range of the **a** is also reduced by **f**. In other words, **a**is a random variable in the interval of $[-2\mathbf{f}, 2\mathbf{f}]$, whereas the value of **f** reduces from 2 to 0 in the period of iterations. While the random values of **a** lie in the range of $[-1, 1]$, the next location of a search agent can be in any position between its current position and the position of the prey. Fig. 5 shows that the inequality $|\mathbf{a}| < 1$ forces the wolves to attack the prey.

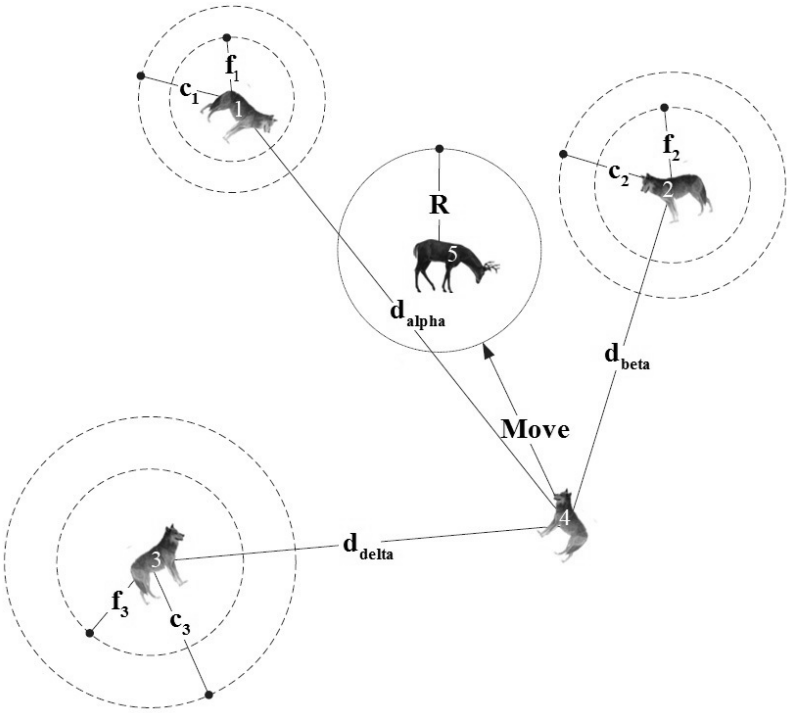

Fig. 5 *Updating the position in GWO algorithm [28].*

According to the operators that have already been presented, GWO algorithm allows the agents to update their positions according to the positions of alpha, beta and delta wolves and attack the prey. Yet GWO algorithms may still be at the risk of trapping in local minimums, so other operators are required to avoid this issue. Although, the proposed encircling mechanism somehow shows identification process, GWO algorithm requires more operators to manifest exploration properties.

3.2.5 Searching for pray

The search process among the GWs is mainly done considering the location of alpha, beta and delta wolves. They fall apart to seek for the prey and aggregate

as they find it. In order to mathematically model the divergence behavior, the **a** vector with random value bigger than 1 or smaller than -1 is used, so that the search agents are forced to diverge and get distant from prey [31]. This procedure shows the identification process and allows the GWO algorithm to search globally. Fig. 6 shows that the inequality $|\mathbf{a}| > 1$ forces the wolves to scatter in the environment to find a better prey.

Another GWO component that affects the identification process is the value of **c**. As in Eq. (7) **c** vector elements are random variables in the interval of [0,2]. This component provides random weights for hunting process to increase ($\mathbf{c} > 1$) or decrease the effect of prey location in determination of the distance in Eq. (5). It also helps GWO to enhance its stochastic behavior along the optimization process and reduce the chance of trapping in local minimums. It should be noted that **c** does not decrease linearly with respect to **a**. **c** is always needed to generate the random values and execute the identification process not only in the first iteration, but also in the last. This factor is very useful for avoiding local minimums especially in the last iteration. **c** vector is also considered as the influence of the obstacles which prevent wolves from approaching the prey in nature [22]. In general, natural obstacles in path of wolves prevent them from approaching the bait with proper speed. This is the precise expression of the **c** vector effect. Depending on wolf's position, the **c** vector can assign a random weight to prey in order to make the hunt harder or easier.

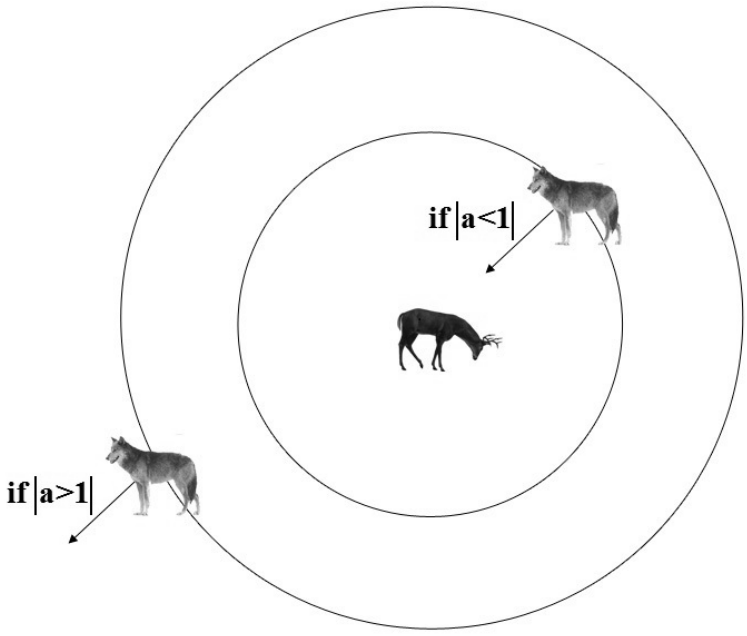

Fig. 6 *Attacking the prey versus searching for the prey.*

In brief, the searching process in GWO algorithm begins with generating a stochastic population of GWs (candidate solutions). During the iteration period,

alpha, beta and delta wolves estimate the possible prey locations. Each candidate solution updates its distance from the prey. The value of **f** is reduced from two to zero, to enhance the process of identifying and attacking the prey. The inequality $|\mathbf{a}| > 1$ results in divergence of the candidate solutions, otherwise they eventually converge toward the prey. Fig. 7 presents the flowchart of GWO algorithm [28].

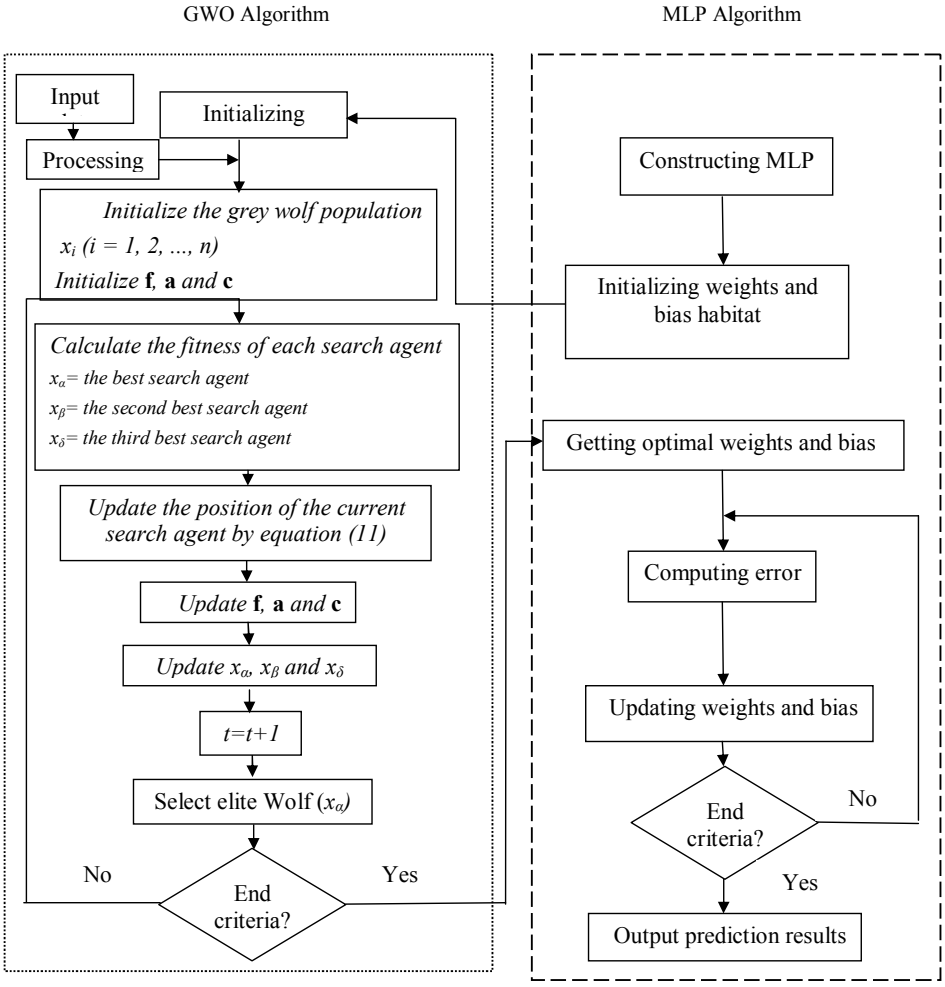

Fig. 7 *Flowchart of GWO algorithm.*

The next section is dedicated to discuss the application of GWO algorithm on MLP NN and its training. The obtained classifier using this learning method is called Gray Wolf Classifier (GWC).

4. Training a Multi-Layer Neural Network using the Gray Wolf Optimization algorithm

In general, there exist three methods for using the EAs for training the MLP NNs. The first method is to use the EAs to find the proper combination of connection weights and node's biases to achieve least possible error in MLP NN. The second method is to use these algorithms to find the suitable structure of MLP NN in a particular problem. The third and last method is to use evolutionary networks to find the parameters of the gradient based learning algorithm such as learning rate and step size. In the present paper, the GWO algorithm is applied to a MLP NN using the first method. In order to design a training algorithm for MLP NNs, the GWO algorithm should have a clear expression of edge weights and node's biases.

In general, there exist three ways of representing connection weights and node's biases: vector, matrix and binary methods [26]. In vector, matrix and binary representation methods, each element is represented by a vector, matrix and string of binary bits, respectively. Each of these representation methods has its own advantages and disadvantages that may be useful in particular cases [24].

Despite the simplicity of transforming the elements to vector, matrix or a string of binary bits in the first method, recovering process of the transformed elements is complicated. So that's the reason this method is often used in NNs. Regarding the second method for networks with complex structures, the data recovery is easier than the coding process. This method is suitable for learning algorithms which are applied in general NNs. The third method requires binary representation of the variables. In this case, the length of each element increases as the complexity of the network increases. Thus the process of coding and decoding is very complicated.

The vector method has been used in this paper since it's not dealing with complex MLP NNs. In order to reduce the execution time of MLP NNs program, general toolbox of Matlab software will not be used. As an example of this coding method, the final vector of MLP NN shown in Fig. 8 is given by Eq. (12):

$$\text{Position } = [w_{13} w_{14} \dots w_{56} w_{57} b_1 b_2 b_3 b_4 b_5] \,. \tag{12}$$

5. Discussion

This section discusses the performed experiments and outlines the test results. Three data sets shown in Tab. I are applied to the classifier which is designed using the GWO algorithm. The data sets are chosen randomly with different dimensions to have a comprehensive evaluation of algorithm. The performance of GWC is compared to three algorithms considering the classification accuracy, local optima avoidance and convergence speed. These algorithms are PSO which is based on SI, Gravitational Search Algorithm (GSA) which is inspired by physics and is based on gravitational force between objects and PSOGSA which is a combination of previous algorithms. The ability of PSO's social thinking and local search of GSA are utilized to increase the convergence speed of the PSOGSA algorithm. The parameters of these algorithms are presented in Tab. II. PSOGSA algorithm has a combination of PSO and GSA parameters. So, they are not mentioned in Tab. II.

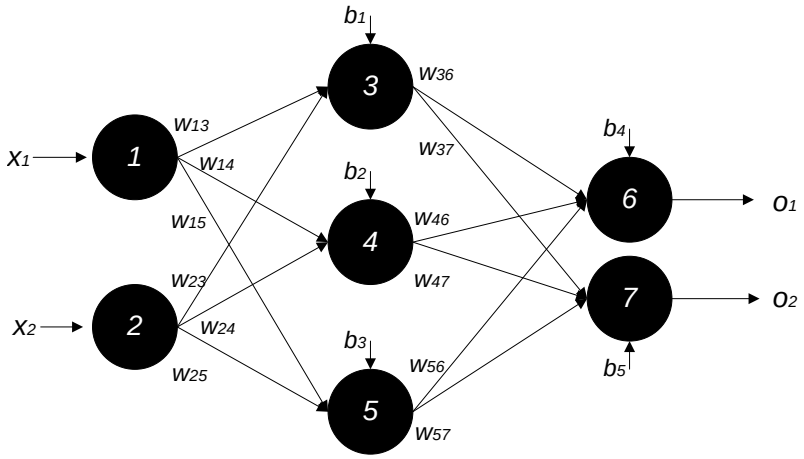

Fig. 8 *MLP NN with (2,3,1) structure.*

Name	Data type	Default task	Attribute characteristics	# Attributes	# Instances	Year
Iris	Classification	Multivariate	Real	4	150	1988
Lenses	Classification	Multivariate	Categorical	4	24	1990
Sonar	Classification	Multivariate	Real	60	208	1988

Tab. I *Data sets used in this paper.*

Algorithm	Parameter	Value
	Topology	Fully connected
PSO	Cognitive constant (c1)	1
	Social constant (c2)	1
	Inertia constant (w)	0.3
	Maximum number of iterations	250
	Population size	200
	Number of masses	30
GSA	Gravitational constant	1
	Maximum number of iterations	250
	Number of wolfs	12
GWC	Lower bound	−5
	Upper bound	5
	Maximum number of iterations	250

Tab. II *Parameters and initial values of the applied algorithms.*

Each network was tested 10 times. The best NN trained so far is chosen for comparison purposes. To do a fair comparison, all algorithms are stopped when the maximum number of iterations reaches 250. Since there is no established standard for choosing the number of hidden nodes in classification of the data sets, based on the structure of MLP NNs, the proposed method in [26] and Eq. (13) are used:

$$h = 2 \times n + 1, \tag{13}$$

where n represents the number of inputs and h indicates the number of hidden nodes. We calculated the average (AVE) and standard deviation (STD) that are observable in the table of results. The ability of algorithms to avoid local minima is shown by these two measures. To reach a greater capability of the algorithm to avoid local minima, the value of AVE $\pm$ STD must be lower. AVE shows the average of MSE over 10 runs and by reaching a lower value for AVE, the greater capability of the algorithm to avoiding local optima and finding solutions near the global optimum is indicated. Averages value of the two algorithms can be equal although their performance in finding the global optimum in each run is different. Because of these reasons, AVE is not a good parameter alone and another parameter like STD will help to specify the dispersion of results. To have a lower dispersion of results, the STD must be lower. So, we can show the ability of an algorithm in avoiding local minima by adding the two mentioned parameter together (AVE $\pm$ STD). Note that the best results are highlighted in bold type in Tabs. III–V.

According to Derrac et al. [5], statistical tests are needed to have an adequate evaluation of performance of heuristic algorithms. Comparing algorithms according to their mean and standard deviation values is not enough [8] and a statistical test is needed to demonstrate a remarkable improvement of a new algorithm in comparison to the other existing algorithms to solve a particular problem [8]. In order to see whether the results of GWO differ from PSO, GSA and PSOGSA in a statistically significant way, a non-parametric statistical test, Wilcoxon's rank-sum test [44], was accomplished at 5% significance level. The calculated p-values in the Wilcoxon's rank-sum are given in the results as well. In the tables, N/A demonstrates "Not Applicable" which means that the corresponding algorithm cannot be compared with itself in the rank-sum test. Conventionally, p-values less than 0.05 are considered as strong evidence against the null hypothesis. Note that p-values greater than 0.05 are underlined in the tables. Another comparative measure shown in the results is classification rates.

5.1 Iris classification problem

There are 150 samples in the Iris dataset that contains three classes: Setosa, Versicolor and Virginica. The features of these samples are sepal length, sepal width, petal length and petal width [13]. MLPs solved this dataset with the structure 4-9-3. Tab. III shows the results of training algorithms to solve this dataset. The results shows GWO and PSOGSA have a better performance to avoid local minima in comparison to the other algorithms according to the results of AVE, STD, and p-values. In addition, percentage of classification for the samples was about 98.6667% that was better than the other algorithms. The convergence curves for

this dataset is shown in Fig. 9 that demonstrates the convergence curve of GWO is better than the other algorithms.

Algorithm	MSE (AVE ± STD)	p-values	Classification rate [%]
GWO	0.0449 ± 0.0269	N/A	98.6667
PSO	0.0869 ± 0.1019	0.0960	93.8666
GSA	0.0542 ± 0.1037	8.6974e-14	90.7999
PSOGSA	0.0427 ± 0.1064	0.0039	96.9333

Tab. III *Experimental results for Iris dataset.*

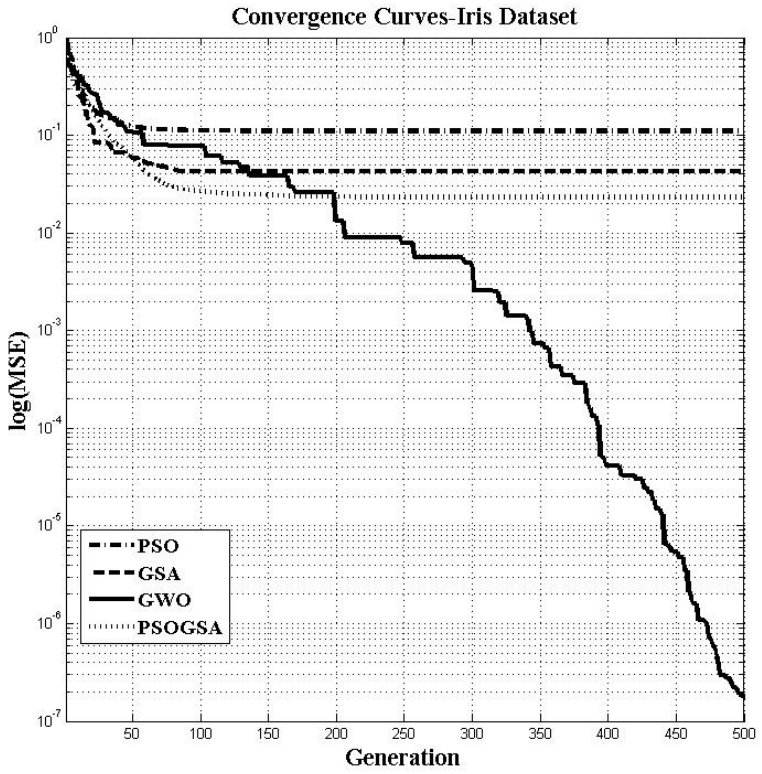

Fig. 9 *Convergence curves of algorithms for Iris dataset.*

5.2 Lenses classification problem

The Lenses dataset tries to predict whether a person will need soft contact Lenses, hard contact Lenses or no contacts, by determining relevant attributes of the client. The dataset has 4 attributes (age of the patient, spectacle prescription, notion on

astigmatism, and information on tear production rate) plus an associated three-valued class, which gives the appropriate lens prescription for patient (hard contact Lenses, soft contact Lenses, no Lenses). The data set contains 24 instances (class distribution: hard contact Lenses (4 instances), soft contact Lenses (5 instances), no contact Lenses (15 instances)) [13]. Tab. IV and Fig. 10 show the results of training algorithms to solve this dataset.

Algorithm	MSE (AVE ± STD)	p-values	Classification rate [%]
GWO	0.0677 ± 0.0489	N/A	98.4328
PSO	0.1956 ± 0.0881	3.0591e-10	97.4123
GSA	0.2768 ± 0.1310	1.0183e-21	95.4112
PSOGSA	0.1847 ± 0.1043	6.2167e-15	99.1666

Tab. IV *Experimental results for Lenses dataset.*

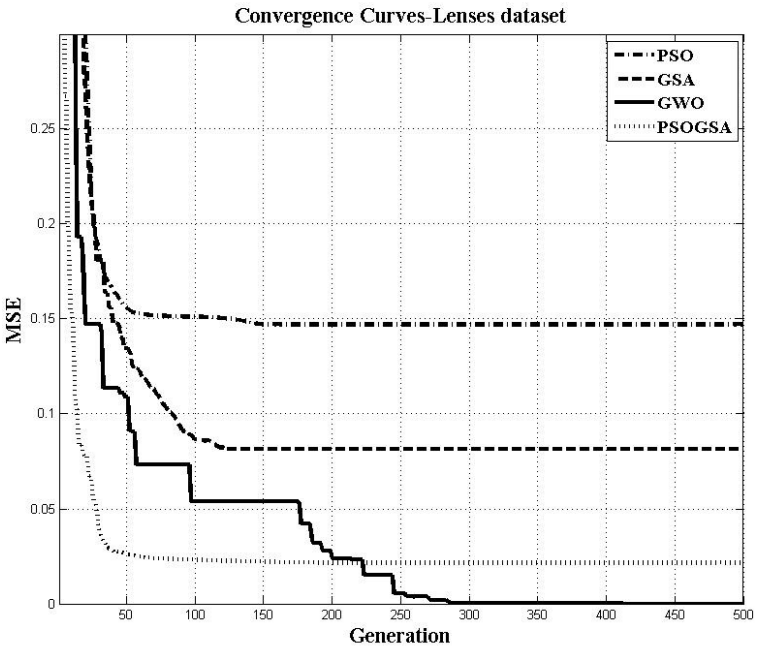

Fig. 10 *Convergence curves of algorithms for Lenses dataset.*

According to the Tab. IV, GWO has the best performance according to the three measures (STD, AVE and p-value). So, it has also a good ability to solve these kind of datasets (dataset with few samples) in terms of avoiding local minima more than the other algorithms. Tab. IV shows that PSOGSA has the most accuracy to classify. This is because of combining the exploration ability of GSA algorithm and the exploitation ability of PSO ability. It is notable that PSOGSA have a

good performance on datasets with low samples. In this case classification rate of GWO algorithm becomes second in the ranking. On the other hand, as you can see in Fig. 10, the convergence rate of GWO algorithm is better than the other algorithms.

5.3 Sonar classification problem

5.3.1 Sonar dataset

Another dataset used in this paper have been extracted from Gorman and Sejnowski test in references [9, 13]. In this test, a metal cylinder (target) with a length of 5 feet and a rock (clutter) with the same size have been put in the sea bed and a wide-band linear FM chirp (ka = 55/6) was sent to them. Transmitted pulse and its spectrogram are shown in Fig. 11. Returned echoes have been collected within 10 yards of them. Based on the SNR of 1200 received echoes, 208 of them which had SNR between 15dB and 4dB have been chosen. From these 208 echoes, 111 of them were from the metal cylinder and 97 of them were from the rock. Fig. 12 shows samples of echoes from rock and metal cylinder.

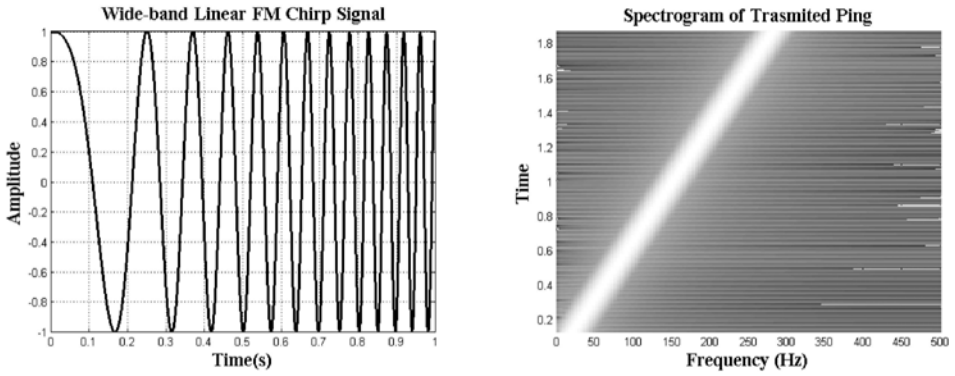

Fig. 11 *Transmitted pulse and its spectrogram.*

According to Fig. 12, returned echoes of the rock and metal cylinder are very similar, so classification becomes hard in this case. Typical networks have a low classification and convergence rate for this dataset. To solve the first problem (low classification rate), we use GWO to train my MLP NN. However, combination of NN and meta-heuristic algorithms beside high dimension of sonar dataset will cause more complexity and redundancy. To solve this problem, we reduced the dimension of sonar dataset from 60 to 9 (number 9 was obtained experimentally). The method to reduce dimension of sonar dataset will be explained in the Section 5.3.2 briefly. Also, the results of applying different classifiers are shown in Fig. 13 and Tab. V.

5.3.2 Dimensionality reduction using principal component analysis

Principal Component Analysis (PCA) is a linear method to reduce data dimensions. This method converts the data into a subspace with a lower dimension. In

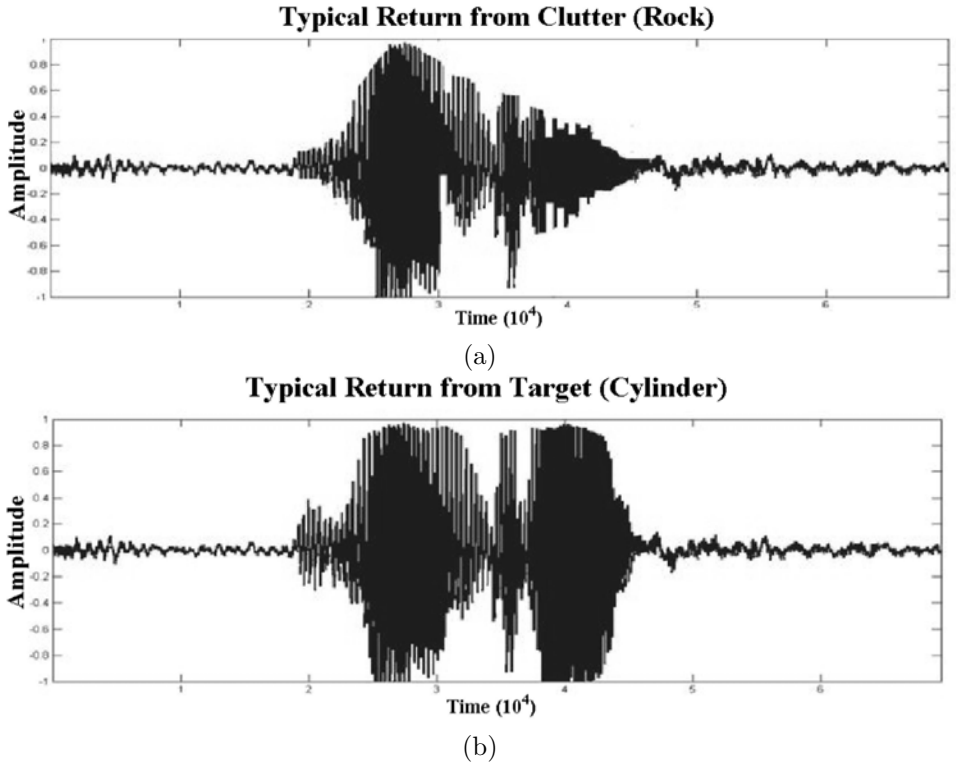

Fig. 12 *Samples of echoes from rock (clutter) and metal cylinder (target).*

fact, this method rotates the coordinate axes in such a way that data with the greatest variance will lie on the main axes. Although many methods have been introduced to reduce the dimensions in recent years, we used classic unsupervised PCA because of its simplicity and low computational complexity. As it is proved in reference [12], new hybrid methods have a better performance comparing to classic methods on virtual datasets. However, classic methods like PCA have a better performance on real datasets. On the other hand, there are two ways to evaluated PCA performance: supervised evaluation and unsupervised evaluation. Supervised evaluation method is highly efficient in terms of accuracy; however, it is very costly in terms of time. Because of the weakness of this method, we used the unsupervised evaluation method in this paper.

PCA tries to find a linear map called $\mathbf{M}$ in order to maximize the cost function defined by Eq. (12):

$$\text{trace}(\mathbf{M}^{\text{T}} cov(\mathbf{X})\mathbf{M}), \tag{14}$$

where $\text{cov}(\mathbf{X})$ is the covariance matrix. It can be demonstrated that this linear transformation is formed by d main feature vectors of the covariance matrix with a zero mean. So, this kind of PCA tries to solve Eq. (13):

$$\text{cov}(\mathbf{X})\mathbf{M} = \lambda\mathbf{M}. \tag{15}$$

d numbers of eigenvalues (λ) can be achieved by solving this equation. Main data $\mathbf{X}$ with b dimensions (in this paper $b = 60$) are turned to a new data $\mathbf{Y}$ with $d = 9$ ($d < b$) dimension by transformation $\mathbf{M}$ and use of Eq. (14):

$$\mathbf{Y} = \mathbf{XM}. \tag{16}$$

5.3.3 Experimental results for sonar dataset

After reducing data dimensions to 9, the new datasets are applied to different classifier. MLP network structure for these datasets is 9-19-2. The test results are shown in Fig. 13 and Tab. V.

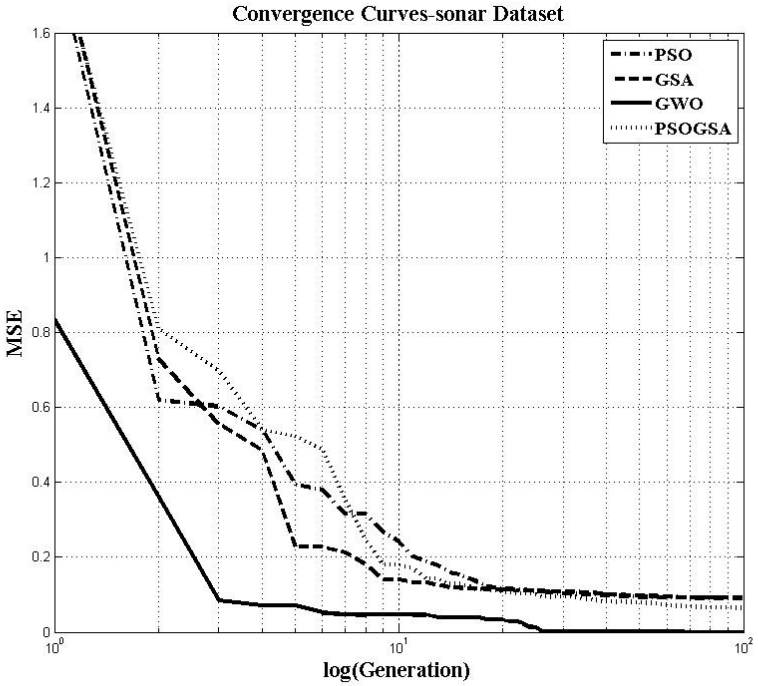

Fig. 13 *Convergence curves of algorithms for Sonar dataset.*

Algorithm	MSE (AVE ± STD)	p-values	Classification rate [%]
GWO	0.0794 ± 0.0112	N/A	95.76922
PSO	0.1311 ± 0.1076	7.2239e-04	93.6741
GSA	0.1049 ± 0.0965	9.2798e-20	92.7500
PSOGSA	0.0850 ± 0.0960	1.0668e-10	94.42308

Tab. V *Experimental results for Sonar dataset.*

According to the Tab. V, GWO has the best performance according to the three measures (STD, AVE and p-value). So GWO classifier has a good ability

to solve these kind of datasets in terms of avoiding local minima and it also has the best classification rate. As can be seen in Fig. 13, this algorithm also has the best convergence rate like the other examples. This results show that this classifier (GWC) has the best performance in classifying Sonar dataset.

6. Conclusions

Poor results of the GSA algorithm that is caused by slowness of its masses in exploitation phase and also the weakness of PSO algorithm in exploration phase have partially fixed by PSOGSA algorithm which combines the abilities of mentioned algorithms. However, there has not been established a good balance between exploitation and exploration phase yet. The GWO algorithm offered a good balance using four types of wolves with different rank and their hunting mechanism. Therefore, we used this algorithm to train the MLP network in this paper. To test the performance of the designed classifier, we applied three different datasets to it including: Iris (with normal dimensions and samples), Lenses (with few samples) and Sonar (with high dimensions). The results demonstrated that the designed classifier has a better or comparable performance in terms of classification rate, convergence rate and local minima avoidance in comparison to the classifiers designed by PSO, GSA and PSOGSA algorithms.

In future studies, it would be interesting to investigate the efficiency of GWO in training other types of NNs such as recurrent, Kohonen, or RBF networks. In addition, employing GWO to define the structure of MLPs is worth investigating.

References

[1] ABEDIFAR V., ESHGHI M., MIRJALILI S., MIRJALILI S.M. An optimized virtual network mapping using PSO in cloud computing. *21st Iranian Conference on Electrical Engineering,* Mashhad, Iran. Mashhad: 2013, pp. 1−6, doi: 10.1109/IranianCEE.2013.6599723.

[2] AUER P., BURGSTEINER H., MAASS W. A learning rule for very simple universal approximators consisting of a single layer of perceptrons. *Neural Networks*. 2008, 21(5), pp. 786−795, doi: 10.1016/j.neunet.2007.12.036.

[3] BARAKAT M., LEFEBVRE D., KHALIL M., DRUAUX F., MUSTAPHA O. Parameter selection algorithm with self adaptive growing neural network classifier for diagnosis issues. *International Journal of Machine Learning*. 2013, 4(3), pp. 217−233, doi: 10.1007/s13042-012-0089-5.

[4] CSÁJI B.C. *Approximation with artificial neural networks*. Hungary, 2001. MSc thesis, Faculty of Sciences Eötvös Loránd University in Hungary, Available from: https://scholar.google.com/citations?view_op=view_citation{&}hl=en{&}usergcRJf_gAAAAJ{&}citation_for_view=gcRJf_gAAAAJ:IjCSPb-OGe4C

[5] DERRAC J., GARCÍA S., MOLINA D., HERRERA F., A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*. 2011, 1(1), pp. 3−18, doi: 10.1016/j.swevo.2011.02.002.

[6] DORIGO M., BIRATTARI M., STUTZLE T. Ant colony optimization. *IEEE Computational Intelligence Magazine*. 2006, 1(4), pp. 28−39, doi: 10.1109/MCI.2006.329691.

[7] EMARY E., ZAWBAA H.M., GROSAN C., HASSENIAN A.E. Feature subset selection approach by gray-wolf optimization. *Afro-European Conference for Industrial Advancement,* Addis Ababa, Ethiopia. Addis Ababa: Springer, 2015, pp. 1−13, doi: 10.1007/978-3-319-13572-4_1.

[8] GARCÍA S., MOLINA D., LOZANO M., HERRERA F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics*. 2009, 15(6), pp. 617−644, doi: 10.1007/s10732-008-9080-4.

[9] GORMAN R.P., SEJNOWSKI T.J. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*. 1988, 1, pp. 75−89, doi: 10.1016/0893-6080(88)90023-8.

[10] GUO Z.X., WONG W.K., LI M. Sparsely connected neural network-based time series forecasting. *Information Sciences*. 2012, 193, pp. 54−71, doi: 10.1016/j.ins.2012.01.011.

[11] HO Y.C., PEPYNE D.L. Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications*. 2002, 115(3), pp. 549−570, doi: 10.1109/cdc.2001.980896.

[12] HOI S.C., JIN R., ZHAO P., YANG T. Online multiple kernel classification. *Machine Learning*. 2013, 90(2), pp. 289−316, doi: 10.1007/s10994-012-5319-2.

[13] http://archive.ics.uci.edu/ml/datasets.

[14] HUSH D.R., HORNE B.G. Progress in supervised neural networks. *IEEE Signal Processing Magazine*. 1993, 10(1), pp. 8−39, doi: 10.1109/79.180705.

[15] ILONEN J., KAMARAINEN J.K., LAMPINEN J. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*. 2003, 17(1), pp. 93−105, doi: 10.1023/A:1022995128597.

[16] KARABOGA D., GORKEMLI B., OZTURK C., KARABOGA N. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*. 2014, 42(1), pp. 21−57, doi: 10.1007/s10462-012-9328-0.

[17] KENNEDY J., EBERHART R. Particle swarm optimization. *IEEE International Conference on Neural Networks,* Perth, Australia. Perth: IEEE, 1995, 4, pp. 1942−1948. Available from: http://link.springer.com/referenceworkentry/10.1007{%}2F978-0-387-30164-8_630{#}page-1.

[18] KIRKPATRICK S., GELATT C.D., VECCHI M.P. Optimization by simulated annealing. *Journal of Science*. 1983, 220(4598), pp. 671−680. Available from: http://link.springer.com/article/10.1007/BF01009452{#}page-1.

[19] LIN L., GEN M. Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation. *Soft Computing*. 2009, 13(2), pp. 157−168, doi: 10.1007/s00500-008-0303-2.

[20] MADADI A., MOTLAGH M.M. Optimal control of DC motor using grey wolf optimizer algorithm. *Technical Journal of Engineering and Applied Sciences*. 2014, 4(4), pp. 373−379.

[21] MAGOULAST G.D., VRAHATIS M.N., Androulakis G.S. On the alleviation of the problem of local minima in back-propagation. *Nonlinear Analysis, Theory, Methods & Applications*. 1997, 30(7), pp. 4545−4550, doi: 10.1016/s0362-546x(96)00369-0.

[22] MECH L.D. Alpha status, dominance, and division of labor in wolf packs. *Canadian Journal of Zoology*. 1999, 77(8), pp. 1196−1203, doi: 10.1139/z99-099.

[23] MIRJALILI S. The ant lion optimizer. *Advances in Engineering Software*. 2015, 83, pp. 80–98, doi: 10.1016/j.advengsoft.2015.01.010.

[24] MIRJALILI S., HASHIM S.Z.M. A new hybrid PSOGSA algorithm for function optimization. *International Conference on Computer and Information Application,* Tianjin, China. Tianjin: ICCIA, 2010, pp. 374−377, doi: 10.1109/iccia.2010.6141614.

[25] MIRJALILI S., HASHIM S.Z.M., SARDROUDI H.M. Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation*. 2012, 218(22), pp. 11125−11137, doi: 10.1016/j.amc.2012.04.069.

[26] MIRJALILI S., LEWIS A. S-shaped versus v-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation*. 2013, 9, pp. 1–14, doi: 10.1016/j.swevo.2012.09.002.

[27] MIRJALILI S., MIRJALILI S.M., HATAMLOU A. Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications*. 2015 in press, doi: 10.1007/s00521-015-1870-7.

[28] MIRJALILI S., MIRJALILI S.M., LEWIS A. Grey wolf optimizer. *Advances in Engineering Software*. 2014, 69, pp. 46−61, doi: 10.1016/j.advengsoft.2013.12.007.

[29] MIRJALILI S.A, MIRJALILI S.M, LEWIS A. Let a biogeography-based optimizer train your multi-layer perceptron. *Journal of Information Sciences*. 2014, 269, pp. 188−209, doi: 10.1016/j.ins.2014.01.038.

[30] MUANGKOTE N., SUNAT K., CHIEWCHANWATTANA C.S. An improved grey wolf optimizer for training q-gaussian radial basis functional-link nets. *International Conference on Computer Science and Engineering (CSEN 2014)*, Bangalore, India. 2014, pp. 209−214, doi: 10.1109/icsec.2014.6978196.

[31] MURO C., ESCOBEDO R., SPECTOR L., COPPINGER R.P. Wolf-pack (canis lupus) hunting strategies emerge from simple rules in computational simulations. *Behavioural Processes*. 2011, 88(3), pp. 192−197, doi: 10.1016/j.beproc.2011.09.006.

[32] NG S.C., CHEUNG C.C., LEUNG S.H., LUK A. Fast convergence for backpropagation network with magnified gradient function. *IEEE Joint Conference on Neural* Networks, Portland, OR, USA. 2003, 3, pp. 1903−1908, doi: 10.1109/ijcnn.2003.1223698.

[33] NGUYEN L.S., FRAUENDORFER D., MAST M.S., GATICA-PEREZ D. Hire me: computational inference of hirability in employment interviews based on nonverbal behavior. *IEEE Transactions on Multimedia*. 2014, 16(4), pp. 1018−1031, doi: 10.1109/tmm.2014.2307169.

[34] OJA E. Unsupervised learning in neural computation. *Theoretical Computer Science*. 2002, 287(1), pp. 187−207, doi: 10.1016/S0304-3975(02)00160-3.

[35] PEREIRA L.A., AFONSO L.C., PAPA J.P., VALE Z.A., RAMOS C.C., GASTALDELLO D.S., SOUZA A.N. Multilayer perceptron neural networks training through charged system search and its application for non-technical losses detection. *IEEE PES Conference on, Innovative Smart Grid Technologies Latin America (ISGT LA)*, Sao Paulo, Brazil. 2013, pp. 1–6, doi: 10.1109/isgt-la.2013.6554383.

[36] RATHEE P., GARG R., MEENA S. Using grey wolf optimizer for image registration. *International Journal of Advance Research in Science and Engineering*. 2015, 4(4), pp. 360−364.

[37] REED R.D., MARKS R.J. Neural smithing: supervised learning in feedforward artificial neural networks. *MIT Press*. 1998, Available from: https://www.amazon.com/Neural-Smithing-Supervised-Feedforward-Artificial/dp/0262527014.

[38] SALIMI H. Stochastic Fractal Search: A powerful metaheuristic algorithm. *Knowledge-Based Systems*. 2015, 75, pp. 1−18, doi: 10.1016/j.knosys.2014.07.025.

[39] SLOWIK A., BIALKO M. Training of artificial neural networks using differential evolution algorithm. *International Conference on Human System Interactions*, Kraków, Poland. Kraków: IEEE, 2008, pp. 60−65, doi: 10.1109/hsi.2008.4581409.

[40] SONG H.M., SULAIMAN M.H., MOHAMED M.R. An application of grey wolf optimizer for solving combined economic emission dispatch problems. *International Review on Modelling and Simulation*. 2014, 7(5), pp. 838−844, doi: 10.15866/iremos.v7i5.2799.

[41] SULAIMAN M.H., MUSTAFFA Z., MOHAMED M.R., ALIMAN O. Using the gray wolf optimizer for solving optimal reactive power dispatch problem. *Applied Soft Computing*. 2015, 32, pp. 286−292, doi: 10.1016/j.asoc.2015.03.041.

[42] WANG P., YU X., LU J. Identification and evolution of structurally dominant nodes in protein-protein interaction networks. *IEEE Transactions on Biomedical Circuits and Systems*. 2014, 8(1), pp. 87−97, doi: 10.1109/tbcas.2014.2303160.

[43] WHITLEY D. A genetic algorithm tutorial. *Statistics and Computing*. 1994, 4(2), pp. 65−85, doi: 10.1007/bf00175354.

[44] WILCOXON F. Individual comparisons by ranking methods. *Biometrics Bulletin*. 1945, 1(6), pp. 80−83, doi: 10.2307/3001968.

[45] ZHANG N. An online gradient method with momentum for two-layer feedforward neural networks. *Applied Mathematics and Computation*. 2009, 212(2), pp. 488−498, doi: 10.1016/j.amc.2009.02.038.