MULTI-OBJECTIVE OPTIMIZATION BY MEANS OF MULTI-DIMENSIONAL MLP NEURAL NETWORKS

Majid Rafei[*], *Samin Ebrahim Sorkhabi*[*], *Mohammad Reza Mosavi*[*]

Abstract: In this paper, a multi-layer perceptron (MLP) neural network (NN) is put forward as an efficient tool for performing two tasks: 1) optimization of multi-objective problems and 2) solving a non-linear system of equations. In both cases, mathematical functions which are continuous and partially bounded are involved. Previously, these two tasks were performed by recurrent neural networks and also strong algorithms like evolutionary ones. In this study, multi-dimensional structure in the output layer of the MLP-NN, as an innovative method, is utilized to implicitly optimize the multivariate functions under the network energy optimization mechanism. To this end, the activation functions in the output layer are replaced with the multivariate functions intended to be optimized. The effective training parameters in the global search are surveyed. Also, it is demonstrated that the MLP-NN with proper dynamic learning rate is able to find globally optimal solutions. Finally, the efficiency of the MLP-NN in both aspects of speed and power is investigated by some well-known experimental examples. In some of these examples, the proposed method gives explicitly better globally optimal solutions compared to that of the other references and also shows completely satisfactory results in other experiments.

Key words: *Artificial neural network, dynamic learning rate, globally optimal solution, multi-objective optimization, multi-dimensional MLP, multivariate activation function*

Received: January 21, 2013 **DOI:** 10.14311/NNW.2014.24.002
Revised and accepted: December 31, 2013

1. Introduction

In the case of large problems with many parameters involved, it is difficult to introduce an efficient regulatory approach to obtain the optimal solutions of the problem. Hence, optimization approaches are investigated in order to complete the design process [1,2]. Some of these approaches consist the use of neural networks

[*]Majid Rafei, Samin Ebrahim Sorkhabi, Mohammad Reza Mosavi – Corresponding Author, Department of Electrical Engineering, Iran University of Science and Technology, Narmak, Tehran 16846-13114, Iran, E-mail: `rafei@elec.iust.ac.ir`, `s_ebrahimi@elec.iust.ac.ir`, `m_mosavi@iust.ac.ir`

(NNs) [3-5], evolutionary algorithms [6-8] and other numerical approaches [9]. Most of the works in NN, which is our main interest in this work, involve recurrent neural networks (RNNs) [5]. The main idea benefited in RNNs as function optimization tools [10] is that the problem is transformed into a set of differential equations. This set of equations is then implemented by the RNN to be solved. So, the structure of the RNN strongly depends on the considered optimization function. On the other hand, due to the dynamic structure of the RNN, network stability should be proved [11]. Examples of the optimization problems solved by the RNNs usually include convex or non-convex spaces with only one minimum. To the best of authors' knowledge, a versatile RNN to solve a broad class of optimization problems has not been reported yet.

Since no comprehensive work has been done to optimize the multi-objective problems by multi-layer perceptron (MLP) NNs, MLPs can be categorized as a new tool for optimization [12]. In the following, we summarize the previously reported notions which have addressed MLP as an optimization framework. In [13], optimization performed by means of Lagrangian method in two steps, first, detecting the optimal values of the decision variables and the second, finding appropriate Lagrange multipliers. Weight updating only depends on the derivatives of the function; therefore, their proposed MLP is not able to solve non-convex problems. In [14] a feed-forward NN (FNN) is utilized to solve the Hamilton-Jacobi non-linear inequality. The energy function of the FNN is an approximation of the Hamilton-Jacobi term and inputs of the network indicate the states of the system. Hence, a feedback path has been constructed between the outputs and the inputs of the network. Starting with a set of initial inputs, the feedback system eventually reaches its steady state and the final outputs are considered as the solutions of the Hamilton-Jacobi equation. Thereby, their proposal is presumably limited to solve the Hamilton-Jacobi term; no effort has been made to extend the method. In [12], in order to solve non-linear programming problems, Reifman and Feldman have used the output of an MLP to provide the decision variables of the objective function. In this work, the optimization has been defined in the framework of a parameter free penalty function. The training algorithm used for the NN is the same as back-propagation (BP) algorithm except for the last layer; in which local gradient is considered to be equal to the sum of the gradients related to the objective and the constraint functions. Nevertheless, in their method, obtaining the global solutions of the problem is not guaranteed.

Our main aim in this paper is to introduce a general method to optimize constrained multi-objective problems. The proposed method is completely based on the regular MLP-NN. This approach does not suffer from the difficulties of the RNNs, i.e. problem dependency and instability issues. In the proposed method, traditional activation functions in the last layer of MLP are replaced with the objective functions. Also, in order to optimize the multivariate functions, multi-dimensional neurons are utilized. In addition, by applying the dynamic learning rate, the network is able to globally search the solution space.

Subsequent sections are arranged as follows. In Section 2., the main idea of using the regular MLP-NN as a tool for solving a univariate function will be introduced. This idea will be generalized to solve/optimize a set of non-linear equations in Section 3. Parameters involved in the algorithm performance will be discussed in

Section 4. Also, in this section, utilized error signals and dynamic learning rate will be described in details. Several single and multi-objective problems are solved in Section 5. in order to investigate the efficiency of the proposed method. Finally, the conclusion will be outlined in Section 6.

2. Basic Concepts: To Solve an Objective Function Using Unidimensional MLP

The ANN in Fig. 1 depicts the structure of a single-input and single-output MLP. The input to the activation function of the second layer in the n-th iteration is obtained by:

$$v_1^{(2)}(n) = \sum_{j=0}^{N^{(1)}} w_{1j}^{(2)}(n) y_j^{(1)}(n) = x(n); y_0^{(1)} = +1. \tag{1}$$

In Eq. (1), $y_j^{(1)}(n)$, for $j = 0, 1, \ldots, N^{(1)}$, designates the output of each neuron in the first layer and comes in the form of:

$$y_j^{(1)}(n) = \varphi_j^{(1)} \left(\sum_{i=0}^{N^{(0)}} w_{ji}^{(1)}(n) p_i \right); j = 1, \ldots, N^{(1)}; N^{(0)} = 1. \tag{2}$$

where p_1 is the input to the ANN and $p_0 = +1$ determines the bias value in the first layer. $w_{kj}^{(l)}(n)$ denotes the connection weight between the neuron k in the layer (l) and the neuron j in the layer (l-1). $\varphi_j^{(l)}$ is the real value univariate activation function of the neuron j in the layer (l) (the layers preceding the last one). Therefore, the output of the ANN is acquired by:

$$y_1^{(2)}(n) = y(n) = f(x(n)). \tag{3}$$

In online training, the energy function for the network of Fig. 1 is expressed in its usual form as:

$$E = \frac{1}{2}(y^* - y)^2 = \frac{1}{2}e^2(x). \tag{4}$$

During the training procedure, algorithm tries to reduce the energy of the network to its minimum possible value; that is to say, as the algorithm proceeds, y approaches $y*$. This is equivalent to solving the real value function $y = f(x)$ in order to reach the desired objective, i.e. y^*.

Hornik in [15] demonstrated that a standard ANN with any partially bounded and non-polynomial activation function is capable of estimating any arbitrary non-linear function to the desired degree of accuracy. Thereafter, in [16] polynomial, rational, Fourier series, and logistic functions for two sets of highly non-linear data, subjected to low and high noises, were evaluated [17-19]. It was demonstrated that networks with polynomial activation functions of the appropriate order, are able to estimate the data of a low noise system with an error equivalent to that of an NN with sigmoid activation functions. Researchers in various studies have corroborated

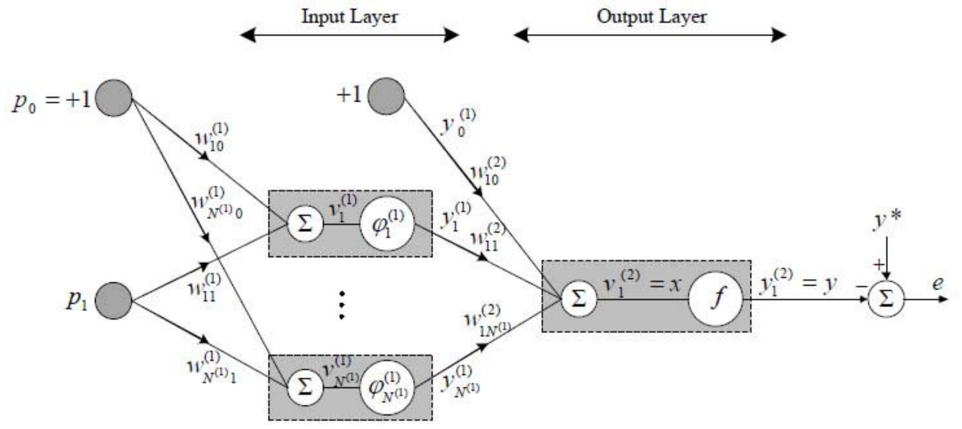

Fig. 1 *The unidimensional perceptron ANN modified for solving a real value objective function f.*

that the polynomial and spline activation functions have the potential to estimate the highly non-linear systems [20-23]. Many other works imply that selection of activation functions is dependent on the application [24,25]. Consequently and as a new point of view, if we choose the arbitrary function $f(x)$ as the activation function in the last layer, so that $f(x)$ would meet the just mentioned conditions, then as the energy of the network approaches its minimum, the output of $f(x)$ will converge to its desired objective. Correspondingly, the input x to function $f(x)$ will converge to the optimal solution $x*$. In other words, the desired function $f(x)$ is solved for the desired objective y^*. Since there are a lot of choices for weights to map a desired MLP input to a desired MLP output, therefore the value of the input to the MLP is of no concern in obtaining the solution of the problem. However, according to experiments, it is preferred to choose the number of the ANN inputs equal to the number of the decision variables, and this relates the network structure to the complexity of the problem.

The desired function to be solved (which is considered as the activation function of the last layer) can be of any aforementioned type, provided to be piecewise continuous and locally bounded. Consequently, *Result 1* is deduced for solving a univariate function.

Result 1: The network in Fig. 1 with any arbitrary input is capable of solving any arbitrary continuous and partially bounded function $f : R \to R$ for the objective $y^* \in R$ within the bounded domain $(x_{\text{low}}, x_{\text{high}})$, assuming that the training algorithm of the network can globally search the solution space.

It is worth mentioning that the limited function $f(x)$ in the last layer implies limited error signal $e(x) = y^* - f(x)$. In MLP network of several hidden layers and univariate activation functions (unidimensional NN), weights adjustment can be properly done during the error BP training procedure [26].

3. Expansion of the Method: To Solve/Optimize a Set of Non-linear Equations Using Multi-dimensional MLP

In the previous section, the basics of employing MLP network to solve a function were brought forward and in this section, it will be expanded for a set of non-linear equations. Suppose that solving the following system of non-linear equations is desired:

$$\mathbf{F}(\mathbf{x}) : \mathbf{R}^m \to \mathbf{R}^s : \begin{cases} f_1(x_1, x_2, \ldots, x_m) - y_1^* = 0 \\ f_2(x_1, x_2, \ldots, x_m) - y_2^* = 0 \\ \vdots \\ f_s(x_1, x_2, \ldots, x_m) - y_s^* = 0 \end{cases} \quad \text{for } \; x_{\text{low}} < x_i < x_{\text{high}}. \tag{5}$$

In the previous section, the solution of $f(x)$ was obtained through considering it as the activation function of a neuron in the last layer. Accordingly, for each function of the system of Eq. (5) a neuron must be considered in the last layer. Since each of the equations in the above system has multiple independent variables, the activation function of the neurons must be of multivariate form. Before realizing the system of Eq. (5) by an MLP, let us briefly discuss the MLPs with multivariate activation functions, which recently proposed by Solazzi and Uncini in [20]. In their literature, it was demonstrated that the proposed multi-dimensional network can be trained with a modified BP algorithm. In their proposed structure, which is depicted in Fig. 2, the number of sigma cells devoted to each activation function is equal to the number of independent variables of that function. In addition, it should be mentioned that in this structure, inputs to an activation function are independent of the inputs to the other functions.

In order to calculate the local gradients of each neuron, partial derivatives of the related activation function with respect to its independent variables should be considered. Suppose that for the network of Fig. 2, $\varphi_i^{(L)}(\mathbf{x}) = f_i(\mathbf{x})$, $\mathbf{v}^{(L)} = \mathbf{x}^{(L)}$, and y_i^* is the objective for the activation function f_i, then the energy function can be defined as:

$$E = \frac{1}{2}\sum_{i=1}^{s} e_i^2(\mathbf{x}) \tag{6}$$

where:

$$e_i(\mathbf{x}) = y_i^* - f_i(\mathbf{x}). \tag{7}$$

Network training [20] is described with:

$$w_{k,ji}^{(l)}(n+1) = w_{k,ji}^{(l)}(n) + \Delta w_{k,ji}^{(l)}(n). \tag{8}$$

The output of each sigma cell is obtained by:

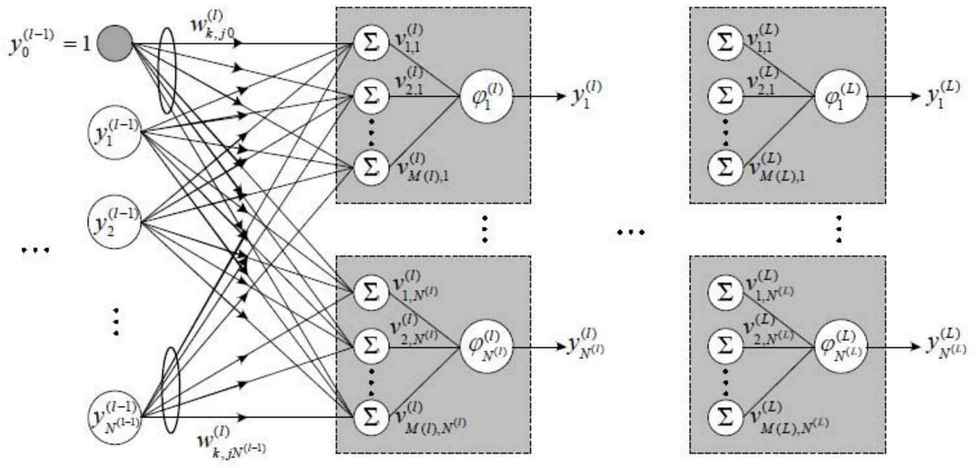

Fig. 2 *The MLP network with multi-dimensional neurons.*

$$v_{k,j}^{(l)} = \sum_{i=0}^{N(l-1)} w_{k,ji}^{(l)}(n) y_i^{(l-1)}(n). \tag{9}$$

The delta rule based on the utilized energy function yields:

$$\Delta w_{k,ji}^{(l)} = -\eta_{k,ji}^{(l)} \frac{\partial E}{\partial w_{k,ji}^{(l)}} = -\eta_{k,ji}^{(l)} \frac{\partial E}{\partial e_j^{(l)}} \cdot \frac{\partial e_j^{(l)}}{\partial y_j^{(l)}} \cdot \frac{\partial y_j^{(l)}}{\partial v_{k,j}^{(l)}} \cdot \frac{\partial v_{k,j}^{(l)}}{\partial w_{k,ji}^{(l)}} = \eta_{k,ji}^{(l)} \delta_{k,j}^{(l)} y_i^{(l-1)} \tag{10}$$

$\eta_{k,ji}^{(l)}$ is the learning rate that is defined for each connection, and the local gradient is:

$$\delta_{k,j}^{(l)}(\mathbf{x}) = e_j^{(l)}(\mathbf{x}) \cdot \frac{\partial \varphi_j^{(l)}(\mathbf{x})}{\partial x_k} = \bigg|_{(\mathbf{x}=\{x_1=v_{1,j}^{(l)} \dots, x_{M(l)}=v_{M(l),j}^{(l)}\})} . \tag{11}$$

The error in each layer is given by:

$$e_j^{(l)}(\mathbf{x}) = \begin{cases} y_j^* - y_j^{(l)} & l = L \\ \sum_{k=1}^{M^{(l)}} \left(\sum_{n=1}^{N^{(l+1)}} \delta_{k,n}^{(l+1)} w_{k,nj}^{(l+1)} \right) & l = L-1, \dots, 1 \end{cases} \tag{12}$$

with $1 \le j \le N^{(l)}, 0 \le i \le N^{(l-1)}$ and $1 \le k \le M(l)$. In the system of Eq. (5), all of the equations have common independent variables. Therefore, for the implementation of the system of Eq. (5) the network of Fig. 2 must be altered to that of Fig. 3, so that a sigma cell will be provided for each of the independent variables of the activation functions. In other words, since the independent variables are the same for all the functions in the system of Eq. (5), the inputs of the multivariate functions are equivalent in the last layer of Fig. 3. Hence, only one common sigma block, consisting of sigma cells, must be applied for all the activation functions. As it is shown in Fig. 3, parallel connections between any sigma cell in the layer (l) and the output of any neuron in the layer (l-1) will be generated. The number of these parallel connections is equal to the number of activation functions in the layer (l). Therefore, in the layer (l) an activation function along with all sigma cells or equivalently the sigma block are considered as a single multi-dimensional neuron, as shown by dashed line. For each of these connections, local gradient would be computed as the product of the error signal and partial derivatives of each of the activation functions in the layer (l) with respect to its independent variables. By considering this, Eqs. (6) to (11), relevant to the BP algorithm, are not changed and are used identically for the training of the network of Fig. 3.

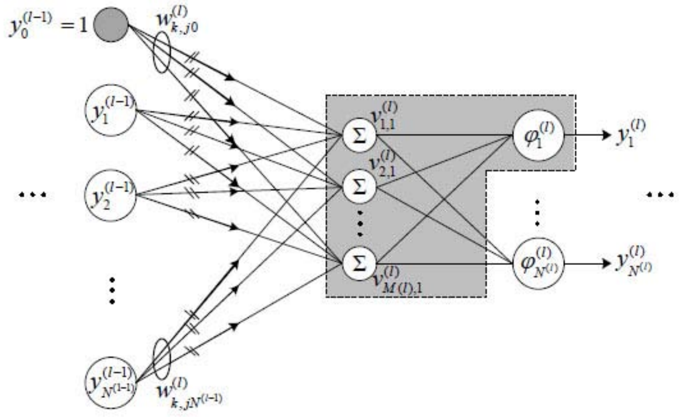

Fig. 3 *The MLP network with multi-dimensional neurons proposed for solving system of non-linear equations in Eq. (5).*

For the goals of this paper, only the last layer contains multivariate activation functions and the preceding layers include univariate activation functions. As mentioned in the previous section, the value of the input to the ANN is of no concern in obtaining the solution of a problem.

Generally, optimization of an objective function f and related constraints g and h is defined as [27]:

$$\text{Minimize} f(x), \text{ subject to } g(x) \geq 0; h(x) = 0. \tag{13}$$

In the optimization of the multi-objective problems including constraints, a neuron will also be considered in the last layer for each of the constraint functions; in this case, by means of the proposed method, only the definition of the error signal has to be modified. If the constraint is violated, its error is proportional to the distance from its boundary. It is noted that the proposed approach is not a penalty method. Because in the proposed method the error and the partial derivatives of the functions, with respect to independent variables, are independently involved in the weight adjustment of the links connected to the corresponding neurons.

In order to explore the design space globally, parameters of the training algorithm should be adjusted properly. In the following section, these parameters are discussed in details.

4. Modification of the Error Signal and Adjustment of the Algorithm Parameters

Depending on the kind of algorithm selected for the training of the network, it is necessary to adjust the effective parameters involved with the training procedure. For the BP algorithm used in the proposed method, several factors are of vital importance; some of these factors are the learning rate, type of the activation functions in hidden layers, error signal of the last layer, energy function, stopping criteria of the algorithm, etc. For instance, if the learning rate is not selected properly, then the algorithm may be trapped in the local minima [28] or may become unstable [29]. Types of the error signals and energy functions are also selected with respect to the type of the data distribution [30-33]. In the following subsections, modifications of the error signal for solving/optimizing a set of non-linear equations and the criteria involved in the proper selection of the parameters are explained in details.

4.1 Modifications on the MLP error signal to solve/optimize a set of non-linear equations

In the proposed method, error signal for the objective functions and the constraints are defined distinctively. For the partially continuous and bounded range functions, error signal will also be bounded. For the unbounded functions, like polynomials that are naturally divergent, it will be required to limit the magnitude of the error signal to a desired interval [34]. Unbounded error signal can be mapped to a bounded space through another function. This mapping function, $u(.)$, non-linearly relates the primary and the secondary space, which is limited in the interval of (a, b). The mapping function $u(.)$ must have the following properties: 1) If $e_1 < e_2$ then $u(e_1) < u(e_2)$ for $e_i \in R$ and 2) $a < u(.) < b$. In fact, to map a ordered set $\mathrm{E} \subset R$ to another ordered set $\mathrm{U} \subset R$, while preserving the order of the members, mapping must be strictly increasing. Real value function $\alpha \,.\, \tanh(x)$, with $x \in R$ and the scale parameter $\alpha \in R$, is one of the most popular squashing functions that limits the range of any unbounded function to the interval $(-\alpha, \alpha)$. As a consequence, we have benefited this function to map unbounded error signals of the objective and constraint functions to bounded ranges.

4.1.1 Error signal for the system of non-linear equations “$\mathbf{F}(\mathbf{x}) = 0$” and the constraints “$\mathbf{H}(\mathbf{x}) = 0$”

Considering the functions with bounded range, customarily, error signal in the output layer for the i-th neuron is defined as:

$$e_i(\mathbf{x}) = y_i^* - f_i(\mathbf{x}) \text{ and } i = 1, \ldots, s \tag{14}$$

in which, y_i^* represents the i-th desired objective value. As previously mentioned, for the unbounded functions, a squashing error signal can be used. Hence, for solving the system of non-linear equations $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_s(\mathbf{x})) = 0$ or for satisfying the constraints of the form $\mathbf{H}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_s(\mathbf{x})) = 0$ with the unbounded range, error signal is redefined as:

$$e_i(\mathbf{x}) = \alpha_i \,.\, \tanh\left(\frac{y_i^* - f_i(\mathbf{x})}{\sigma_i}\right). \tag{15}$$

As $\mathbf{x}$ approaches $\mathbf{x}^*$, the function $f_i(\mathbf{x})$ and the error signal $e_i(\mathbf{x})$ converge to y_i* and zero, respectively. Smoothness of the function $e_i(\mathbf{x})$ alters by adjusting the parameter σ_i. The smaller the parameter σ_i, the more will be the variations of $e_i(\mathbf{x})$ around the origin and therefore, it will be more sensitive to the variations of $f_i(\mathbf{x})$ in the origin. As a rule of thumb, the value of σ_i is selected to be equal to the one third of the maximum predictable value for $f_i(\mathbf{x})$.

4.1.2 Error signal for the set of objective functions in the problem of "min $\mathbf{F}(\mathbf{x})$"

For the set of objective functions $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_s(\mathbf{x}))$, error signal can be defined as:

$$e_i(\mathbf{x}) = y_{i,\min} - f_i(\mathbf{x}) \text{ and } i = 1, \ldots, s \tag{16}$$

where $y_{i,\min}$ is the minimum value of the i-th objective reached by the algorithm up to the present stage of training process, i.e. n, and is stated as:

$$y_{i,\min} = \min\nolimits_{1 \le j \le n} y(j). \tag{17}$$

Eq. (16) implies that the output of the network always tracks the lowest value. By applying the error signal of Eq. (16) in the algorithm, two problems may occur. First, the execution of the algorithm may be stopped easily in the regions where the function $f_i(\mathbf{x})$ is flat or it may be trapped in the local minima. For example, suppose that $f_i(\mathbf{x})$ had the value of 2 in the previous iteration and the value of $y_{i,\min} = 1$ has been reached up to that iteration. Therefore, according to Eq. (16), the error in the previous iteration is equal to -1. If in the present iteration a value between 1 and 2 is reached for $f_i(\mathbf{x})$, then the value of its error signal is smaller than the previous one. Since, the objective is to minimize the absolute value of the error, based on the energy function defined in Eq. (7), the weight variations become smaller as the iterations proceed. As a result, iterations may continue around $y_{i,\min} = 1$ and ultimately the algorithm will stop; whereas, it is probable that the global minimum may be in a value smaller than 1. The second problem arises when a value very smaller than $y_{i,\min}$ is achieved, thereby, the weights may vary drastically. For example, if the objective function has a global minimum that lies in a deep narrow valley, then falling into this minimum, the error value may encounter severe instantaneous variations. Therefore, because of the intense fluctuations of weights the minimum may be lost.

As an approach to overcome the two problems stated, it is proposed to utilize the squashing mapping function:

$$e_i(\mathbf{x}) = \alpha_i \,. \left(\tanh \left(\frac{\varepsilon_i(\mathbf{x})}{\sigma_i} + \theta_i \right) + 1 \right) \tag{18}$$

$$\varepsilon_i(\mathbf{x}) = f_i(\mathbf{x}) - y_{i,\min} \text{ and } i = 1, \ldots, s \tag{19}$$

Eq. (18) is displayed in Fig. 4 for $\alpha_i = 0.5$. In this function, α_i is the maximum value of the range, σ_i indicates the smoothness and the error in $f_i(\mathbf{x}) = y_{i,\min}$ determines the value of θ_i. If λ_i represents the error value at the point $f_i(x) = y_{i,\min}$, then the constant θ_i is obtained by:

$$\theta_i = \operatorname{arctanh} \left(\frac{\lambda_i}{\alpha_i} - 1 \right) . \tag{20}$$

As long as $f_i(\mathbf{x})$ is greater than $y_{i,\min}$, the error $e_i(\mathbf{x})$ is greater than λ_i. In the vicinity of the local minimum, if $f_i(\mathbf{x})$ decreases, then the value of the error will be always less than λ_i, otherwise whenever $f_i(\mathbf{x})$ increases, the value of the error will be greater than λ_i. Parameter λ_i should not be selected very small, otherwise, the variations of weights in the vicinity of $y_{i,\min}$ will be very small and accordingly, the progress of the algorithm will be very slow. Indeed the value of λ_i is equivalent to the maximum acceptable error in local minima. σ_i and λ_i are somehow crucial parameters for that they determine the effectiveness of the functions in weight adjustment. For instance, the function $f_i(\mathbf{x})$ considered with a large λ_i or a small parameter σ_i, will be more effective in the weight adjustment; this correspondingly means that the minimization of this objective function is of more importance in comparison with others. Hence, the values of λ_i and σ_i have to be selected according to the importance of each objective function $f_i(\mathbf{x})$.

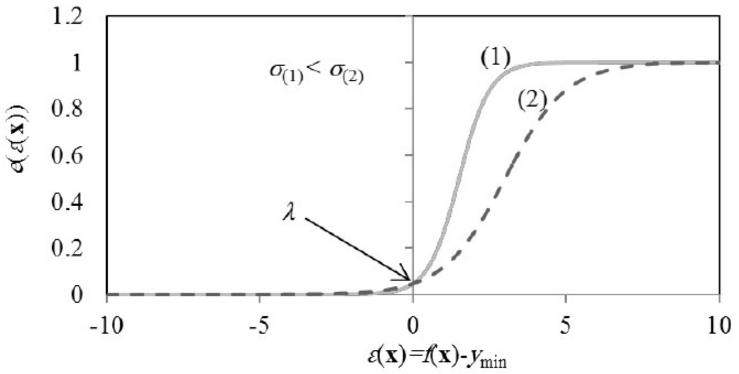

Fig. 4 $e_i(x) = 0.5\,(\tanh(\varepsilon_i(x)/\sigma_i + \theta_i) + 1)$ *as a squashing function of the error signal* $(\varepsilon_i(x))$ *utilized in the optimization algorithm.*

4.1.3 Error signal for the inequality constraints "$\mathbf{G}(\mathbf{x}) \geq 0$"

As explained before, in the inequality constraints (including the boundary constraints) $\mathbf{G}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_t(\mathbf{x}))$ if the constraint is violated, the corresponding error would be proportional to the distance between its current value and the boundary. In the proposed method, the objective is to reach the boundary of the constraints and the internal points where the constraints hold. If a constraint is potentially divergent, then it must be bounded using a squashing function. In this case, the origin of the coordinate corresponds to the boundary of the constraint, i.e. $\mathbf{g}^* = (g_1^*, \ldots, g_t^*) = 0$. Consequently, $\mathbf{g}^* - \mathbf{G}(\mathbf{x}) = -\mathbf{G}(\mathbf{x})$ and:

$$e_j(\mathbf{x}) = \begin{cases} \alpha_j \,.\, \left(\tanh\left(\dfrac{-g_j(\mathbf{x})}{\sigma_j} + \theta_j\right) + 1\right) & \text{for } g_j(\mathbf{x}) < 0 \\ 0 & \text{for } g_j(\mathbf{x}) \geq 0 \end{cases} \quad \text{and } j = 1, \ldots, t. \tag{21}$$

Parameters in this equation are all described before. It is essential to satisfy the constraints prior to the minimization of the objective function $f_i(\mathbf{x})$. Hence, the parameters α_j, σ_j and θ_j for the jth constraint has to be selected so that its error would have more influence in weight adjustment. It should be mentioned that, in the case where we use the error signal of the general form $e_j^{(L)}(\mathbf{x}) = \alpha_j \,.\, (\tanh(\varepsilon_j(x)/\sigma_j + \theta_j) + \xi)$ for the objective functions or the constraints, Eq. (11) should be altered as follows. Assuming that $\varepsilon_j(\mathbf{x}) = y_j^* - f_j(\mathbf{x})$:

$$\delta_{k,j}^{(L)}(\mathbf{x}) = \frac{\alpha_j}{\sigma_j}\left(1 - \left(\frac{e_j^{(L)}(\mathbf{x})}{\alpha_j} - \xi\right)^2\right) .e_j^{(L)}(\mathbf{x}) \,.\, d_j^{(L)}(\mathbf{x}) \tag{22}$$

and with regard to $\varepsilon_j(\mathbf{x}) = f_j(\mathbf{x}) - y_{j,\min}$:

$$\delta_{k,j}^{(L)}(\mathbf{x}) = \frac{-\alpha_j}{\sigma_j}\left(1 - \left(\frac{e_j^{(L)}(\mathbf{x})}{\alpha_j} - \xi\right)^2\right) .e_j^{(L)}(\mathbf{x}) \,.\, d_j^{(L)}(\mathbf{x}) \tag{23}$$

where in both cases:

$$d_j^{(L)}(\mathbf{x}) = \left.\frac{\partial \varphi_j^{(L)}(\mathbf{x})}{\partial x_k}\right|_{(\mathbf{x}=\{x_1=v_{1,j}^{(L)}, \ldots, x_m=v_{m,j}^{(L)}\})} \tag{24}$$

In the above equation, $\varphi_j^{(L)}(\mathbf{x})$ resembles one of the functions $f_j(\mathbf{x})$, $g_j(\mathbf{x})$, or $h_j(\mathbf{x})$. Error signals utilized for optimization/problem solving and constraints are summarized in Tab. I. Ultimately, based on the new concepts brought up in this subsection the following result is concluded.

Result 2: Network in Fig. 3 with any arbitrary input can solve (optimize) the system of non-linear equations in Eq. (5) for the optimal values of $\mathbf{x}^* \in R$, which yields to the desired $\mathbf{y}^* \in R$ (extreme values $\mathbf{F}(\mathbf{x}^*)$), supposing that network training algorithm would be capable to search the solution space globally.

Optimization Function	**Error Signal**
Solving system of non-linear equations: "$\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_s(\mathbf{x})) = 0$"	$e_i(\mathbf{x}) = \alpha_i \,.\, \tanh\left(\dfrac{y_i^* - f_i(\mathbf{x})}{\sigma_i}\right)$
Optimization of functions: "Min $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_s(\mathbf{x}))$"	$e_i(\mathbf{x}) = \alpha_i \,.\, \left(\tanh\left(\dfrac{f_i(\mathbf{x}) - y_{i,\min}}{\sigma_i} + \theta_i\right) + 1\right)$
Equality Constraints: "$\mathbf{H}(\mathbf{x})=(h_1(\mathbf{x}), \ldots, h_r(\mathbf{x}))=0$"	$e_j(\mathbf{x}) = \alpha_j \,.\, \tanh\left(\dfrac{h_j^* - h_j(\mathbf{x})}{\sigma_j}\right)$
Inequality Constraints: "$\mathbf{G}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_t(\mathbf{x})) \geq 0$"	$e_j(\mathbf{x})=\begin{cases}\alpha_j \,.\, \left(\tanh\left(\frac{-g_j(\mathbf{x})}{\sigma_j} + \theta_j\right)+1\right) & \text{for } g_j(\mathbf{x}) < 0 \\ 0 & \text{for } g_j(\mathbf{x}) \geq 0\end{cases}$

Tab. I *Summarizing of the error signals related to optimization/problem solving in the proposed method.*

4.2 Modified dynamic learning rate

The learning rate is one of the most important parameters for the algorithm to search the solution space effectively. It has been proved that the dynamic learning rate would improve the performance of the NN in both aspects of the algorithm speed and global search. In most cases, it is desirable to alter the learning rate in a decreasing manner, as the algorithm proceeds [28]. Learning rates dependent on the error or the derivative of the output function, both have been already exploited [29] and adapted for the goal of global search. Here, we benefit the dynamic learning rate proposed in [28]. It was claimed that with this learning rate the global search of the space is guaranteed. The learning rate introduced in [28] is based upon the step size rule and in the n-th iteration is defined as:

$$\eta_n = \beta_n \,.\, \frac{E(w(n)) - E_n^{\text{lev}}}{C^2} \tag{25}$$

where C is a positive real constant and β_n is a positive random constant. If E^* is the smallest attainable value of the network energy, then E_n^{lev} is an estimation of E^* and it is modified in each iteration based on the observed value for $E(w(n))$. E_n^{lev} is defined as:

$$E_n^{\text{lev}} = E_{\min} - \kappa_n \tag{26}$$

in which:

$$E_{\min} = \min\nolimits_{1\leq i\leq n} E(w(i)). \tag{27}$$

κ_n is an adjustable parameter and have to satisfy the following condition:

$$\kappa_n \to 0 \text{ as } n \to \infty \text{ and } \sum_{n=0}^{\infty} \kappa_n^2 = \infty. \tag{28}$$

In this work, it is selected as:

$$\kappa_n = \rho / \sqrt[\gamma]{n} \tag{29}$$

in which ρ and γ are positive real constants and $\gamma > 2$. With this learning rate, the factor κ_n changes in a diminishing manner as the iterations proceed and the energy value gradually approaches the estimated energy. This implies that the initial global search will be gradually converted to a local search. Note that the dynamic learning rate is applied only for the connections in the last layer so that there is a dynamic learning rate corresponding to each of the activation functions. For the rest of the layers a constant learning rate is considered. The convergence of the network with the learning rate of Eq. (25), has been fully proved in [28]. Despite the high ability mentioned in [28], in some cases algorithm may be trapped in the local minima. For example in the case of function:

$$f(x) = -\exp(-x^2) - 3\exp(-100(x-1.5)^2) + 2.5 \tag{30}$$

which is revealed in Fig. 5, there exist a local minimum with a wide basin of attraction in $x = 0$ beside a global minimum that lies in a deep and narrow valley in $x = 1.5$. Applying the learning rate in Eq. (25), algorithm is often trapped in $x = 0$. In order to improve the network performance, when trapping into a local minimum, it is essential to create a situation so that the algorithm can leave the minimum and resume searching the solution space. If this process iterates for q times, then with high probability the discovered minimum is the global minimum. Employing this idea, the probability of finding the global minimum is increased to a higher degree. The results show that for the mentioned function, the algorithm will be able to easily find the global minimum of the function in Eq. (30).

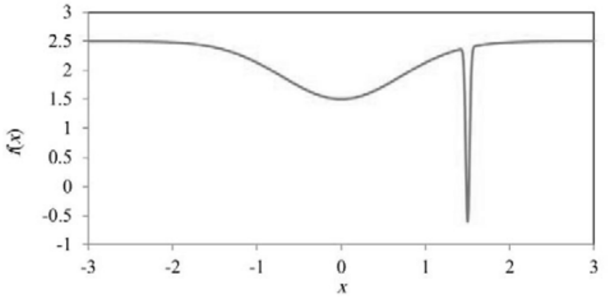

Fig. 5 *The function presented in Eq. (30).*

In the modified learning rate, the essential condition to leave the local minimum is to be informed of the last maximum value. Therefore, as a rule, if n is the number of iterations then it can be expressed that, for an output y, if $y(n-1)$ is the maximum value of the bounded sequence $\{y(n-2), y(n-1), y(n)\}$, then $y(n-1)$ is a local maximum. During the algorithm execution, the values for the network outputs in the current and the two previous iterations are stored. The discovered maximum corresponds to the last maximum value of the network energy, i.e. E_{high}. Even if the exact value of the maximum may not be found because of the tunneling phenomenon, it may suffice to satisfy the goals of the proposed optimization method.

Based on the previous notations, the steps to implement the improved learning rate can be stated as follows. Let $\kappa_n = \rho/\sqrt[3]{n-n'}$, in which the initial value for n' is equal to zero:

- **Step 1** If the variations of the output value in q subsequent iterations is less than the value of Δy_{min}, then the discovered solution in the current iteration n, is a local minimum. Let $E_{\text{low}} = E(n)$ and $n' = n-1$;
- **Step 2** Let the value of E_{min} in Eq. (26) be equal to the difference of the maximum energy, i.e. E_{high}, found with the help of the recently mentioned rule and the current minimum energy value attained, i.e. E_{low}. Hence, we have $E_{\text{min}} = E_{\text{high}} - E_{\text{low}}$;
- **Step 3** Calculate $\kappa_n = \rho/\sqrt[3]{n-n'}$;
- **Step 4** Calculate E^{lev} in Eq. (26);
- **Step 5** Calculate η in Eq. (25).

In the above procedure with the detection of local minimum and previously obtained local maximum, it will be possible to obtain the magnitude of the variations in E^{lev} needed to leave the local minimum, which is equal to the difference of these two energies. Along with this, we update the number of iteration in κ_n so that it starts from one. Hence, with this procedure there will be some conditions, so that the algorithm may resume searching some regions in the vicinity of the last found local minimum. Setting the number of iterations to one and updating the energy E^{lev}, we have again enlarged the steps of the variations for the learning rate in order to continue the search globally. Therefore, we have tried to save the computational overhead up to this stage of the work and make it useful for the rest.

The number of the times that the algorithm has been trapped in a local minimum, the MSE error, and the number of iterations are some of the stopping criteria of the algorithm. The flowchart of the algorithm execution in the proposed multi-objective optimization method is given in Fig. 6.

It is worth mentioning that in the gradient descent rule utilized in the BP algorithm, as the derivative of the error approaches zero in the local minima or maxima, the error will not affect the weight adjustment anymore and as a consequence the algorithm will stop. Due to this reason, in the non-convex problems the algorithm may not leave the local extremes easily. To address this problem, we have revised Eq. (24) as:

$$d_j^{(L)}(\mathbf{x}) = \begin{cases} \left.\dfrac{\partial \varphi_j^{(L)}(\mathbf{x})}{\partial x_k}\right|_{(\mathbf{x}=\{x_1=v_{1,j}^{(L)},\ldots,x_{M(L)}=v_{M(L),j}^{(L)}\})} + v & \text{if} \quad d_j^{(L)} \text{ (in Eq. (24))} \geq 0 \\ \left.\dfrac{\partial \varphi_j^{(L)}(\mathbf{x})}{\partial x_k}\right|_{(\mathbf{x}=\{x_1=v_{1,j}^{(L)},\ldots,x_{M(L)}=v_{M(L),j}^{(L)}\})} - v & \text{if} \quad d_j^{(L)} \text{ (in Eq. (24))} < 0 \end{cases} \tag{31}$$

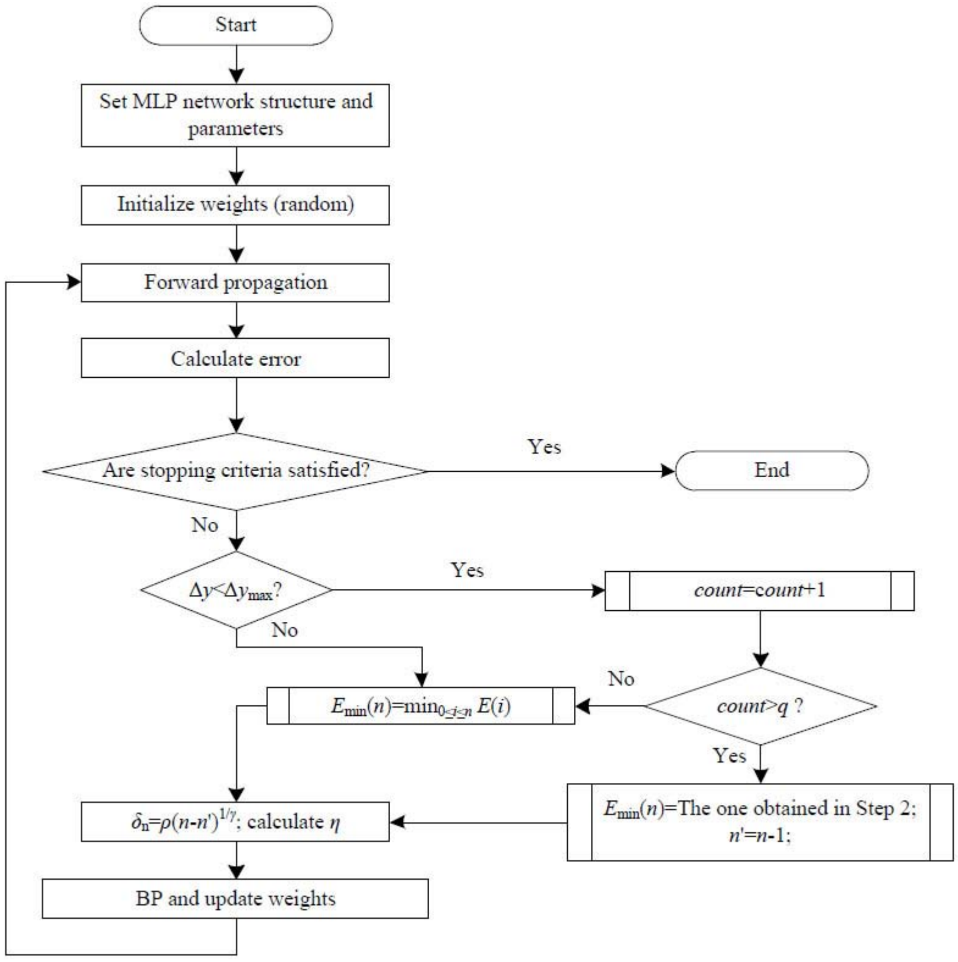
Start
Set MLP network structure and parameters
Initialize weights (random)
Forward propagation
Calculate error
Are stopping criteria satisfied?
Yes
End
No
$\Delta y < \Delta y_{max}$?
Yes
$count = count + 1$
No
No
$E_{min}(n) = \min_{0 \le i \le n} E(i)$
$count > q$?
Yes
$E_{min}(n)$=The one obtained in Step 2; $n' = n - 1$;
$\delta_n = \rho(n - n')^{1/\gamma}$; calculate η
BP and update weights

Fig. 6 *Block diagram of the proposed method for the multi-objective optimization/solving non-linear system of non-linear equations using multi-dimensional MLP network.*

In Eq. (31), ν is a small real constant greater than zero and in this work, it is selected to be one. In the next section, the experimental results will be evaluated. It will be illustrated that, using the proposed method and the modified dynamic learning rate, the minimum in comparison with the literatures, which are referred, are globally discovered.

5. Experimental Results

In this section, in order to examine the efficiency of the proposed method, some examples of several optimization problems are presented and the performance of the network in each case is evaluated. The given examples were all previously investigated in other literatures. The comparison is made in terms of different aspects such as, the speed of convergence to a feasible solution, the ability to reach the optimal solutions in the complicated experimental spaces, and etc. The rate of convergence to the optimal solutions, in the problems that appeared as real time applications and implemented in the hardware form, is of great significance. These problems usually have a convex space or they would be confined to a convex space by some constraints. On the other hand, in incomprehensible problems with unknown spaces, the efficiency of global searching among the various suggested methods is of more concern; these problems usually appear in the fields such as economic and social sciences as well as multi-dimensional problems declared in applied sciences. To perform the calculations a Pentium 4 system of 2.6 GHz CPU and a 4 GB RAM was utilized. The activation function of the neurons in the hidden layers are selected to be logistic $\varphi(x) = 1/(1 + \exp(-ax))$.

Example 1: Convex non-linear programming [35]; the problem is expressed as:

$$\begin{array}{ll} \min & f(\mathbf{x}) = x_1^2 + 2x_1x_2 + x_2^2 + (x_1 - 1)^4 + (x_2 - 3)^4 \\ \text{subject to} & \begin{cases} g_1(\mathbf{x}) = -x_1^2 - x_2^2 + 64 \geq 0 \\ g_2(\mathbf{x}) = -(x_1 + 3)^2 - (x_2 + 4)^2 + 36 \geq 0 \\ g_3(\mathbf{x}) = -(x_1 - 3)^2 - (x_2 + 4)^2 + 36 \geq 0 \\ x_i \geq 0, i = 1, 2. \end{cases} \end{array} \tag{32}$$

The error signals for the function $f(\mathbf{x})$ and the constraints that are all of the inequality form, are given in Eq. (18) and Eq. (21), respectively. The algorithm parameters are tabulated in Tab. II. The network structure is [2,3,4,6], in which 2 indicates the number of the network inputs, 3 and 4 are the size of the first and second hidden layers, and 6 stands for the functions to be optimized, including one objective function and 5 constraints. In the rest of the examples, this method is utilized to present the network structure. The algorithm initiates from the point $\mathbf{x}_0 = (-4, -9)$ and the parameters are $\gamma = 5$, $\rho = 36$, and $C = 4.9$, which C is defined in Eq. (25). Fig. 7 depicts the path through which the algorithm has reached the optimal solution, i.e. $\mathbf{x}^* = (0.0, 1.2)$, with respect to time and the number of iterations. For all of the constraints, the parameters are $\alpha = 0.5$, $\sigma = 7.44$, and $\lambda = 0.03$.

Net. No.	α	σ	λ	$\mathbf{x}^*$	*Constraints*	$f(\mathbf{x}^*)$
Ex. 1	**0.5**	**17.2**	**0.02**	**(0,1.20)**	**(62.6,0.00,0.00)**	**12.94**
[35]	-	-	-	(0,1.96)	(60.2,-8.52,-8.52)†	6.01

†Constraints are not satisfied!

Tab. II *The algorithm parameters for the objective function in Ex. 1 and the results in comparison to [35].*

Example 2: Multi-objective convex non-linear programming [36]; the problem was solved by an extended method of TOPSIS for the convex non-linear multi-objective problems:

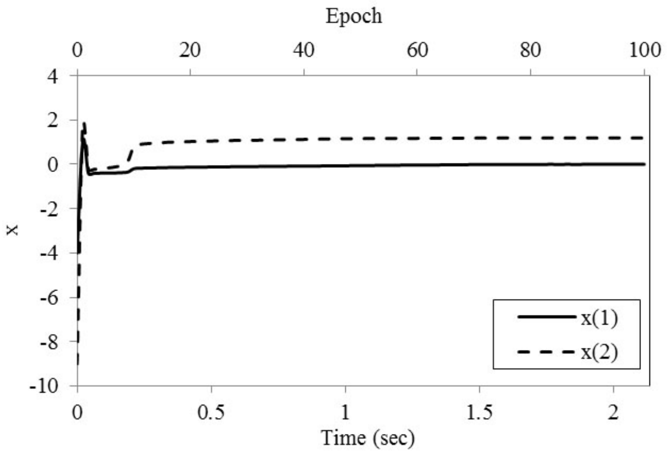

Fig. 7 *Transient performance of the decision variables declared in Example 1.*

$$
\begin{array}{ll}
\max & f_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 \\
\max & f_2(\mathbf{x}) = (x_1-1)^2 + x_2^2 + (x_3-2)^2 \\
\min & f_3(\mathbf{x}) = 2x_1 + x_2^2 + x_3 \\
\text{subject to} & \begin{cases} g_1(\mathbf{x}) = -x_1 + 3x_2 - 4x_3 + 6 \geq 0 \\ g_2(\mathbf{x}) = -2x_1^2 - 3x_2 - x_3 + 10 \geq 0 \\ \mathbf{x} \in R^3 \\ 0 \leq x_1 \leq 3, 0 \leq x_2 \leq 4, 0 \leq x_3 \leq 2. \end{cases}
\end{array}
\tag{33}
$$

The corresponding error signals were given in Tab. I. Since the problem is multi-objective, therefore, the user has to determine the significance of each of the optimization functions. Doing so, the value of each of the parameters, i.e. α, σ, and λ, can be appropriately determined. In order to observe the effect of these three parameters, four cases of the optimization results are listed in Tab. III. Fig. 8 illustrates one of the optimal solutions, i.e. $\mathbf{x}^* = (0.00, 0.05, 0.99)$, with respect to the algorithm parameters in the first row of Tab. III. The selected structure for the network is [3,3,4,11], in which 11 is related to 3 objectives plus 8 constrains. The initial point for all of the decision variables in Tab. III is $\mathbf{x}_0 = (5, 5, 5)$, and γ, ρ and C are 2, 1, and 5, respectively. For the constraints the parameters are selected as: $\alpha = 0.5$, $\sigma = 3$, and $\lambda = 0.1$.

Example 3: Non-convex non-linear programming [37]; considering the problem:

$$
\begin{array}{ll}
\min & f(\mathbf{x}) = |x_1| - x_2^2 \\
\text{subject to} & \begin{cases} g_1(\mathbf{x}) = -|x_1| + 5 \geq 0 \\ g_2(\mathbf{x}) = -2x_2 - x_2^2 + 5 \geq 0 \\ h(\mathbf{x}) = x_1 + x_2 - 1 = 0 \end{cases}
\end{array}
\tag{34}
$$

and the initial point $\mathbf{x}_0 = (6, -2)$, the proposed algorithm converges to the critical point $\mathbf{x}^* = (4.44, -3.44)$, as illustrated in Fig. 9. The network structure is [2,3,4,4]

Net. No.	α	σ	λ	$\mathbf{x}^*$	$\mathbf{F}(\mathbf{x}^*)$
Ours 1	**(0.5,0.5,0.5)**	**(5,20,5)**	**(0.09,0.0002,0.07)**	**(0.00,0.05,0.99)**	**(0.98,2.02,0.99)**
Ours 2	**(0.5,0.5,0.1)**	**(5,20,5)**	**(0.09,0.0002,0.07)**	**(0.00,0.78,0.75)**	**(1.16,3.17,1.36)**
Ours 3	**(0.5,0.5,0.5)**	**(5, 5,5)**	**(0.09,0.0002,0.07)**	**(0.52,0.88,0.00)**	**(1.04,5.03,1.80)**
Ours 4	**(0.5,0.5,0.5)**	**(5,20,5)**	**(0.01,0.0002,0.07)**	**(0.54,0.21,0.00)**	**(0.33,4.26,1.12)**
[36]	-	-	-	(0.00,0.00,1.17)	(1.37,1.69,1.17)

Tab. III *The algorithm parameters for the objective functions in Ex. 2 and the results in comparison to [36].*

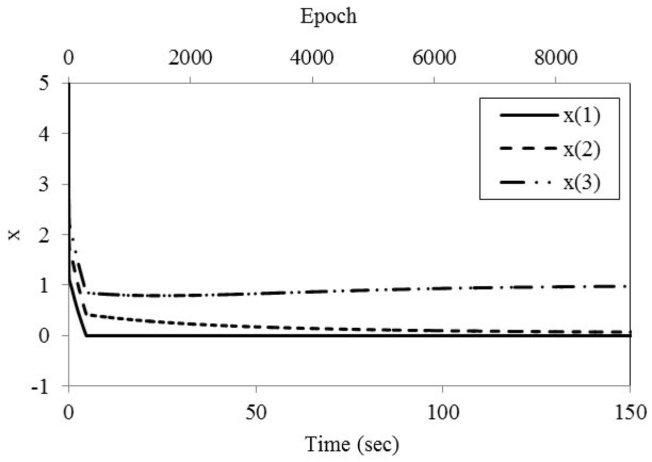

Fig. 8 *Transient response of the independent variables presented in Example 2.*

and the parameters of the proposed algorithm are selected as: $C = 1$, $\gamma = 10$, and $\rho = 10$; and for the constraints the parameters are: $\alpha = (0.5, 0.5, 0.5, 1)$, $\sigma = (7, 7, 3, 7)$, and $\lambda = (0.05, 0.05, 0.07)$. In Tab. IV algorithm parameters for the objective function are listed and the result is given for comparison. Small ripples observed in the flat part of the curve in Fig. 9 take place at the times that the algorithm tries to leave the boundaries of the constraints in order to minimize the objective function. Thereby, due to the violation of the constraints, the algorithm may imply changes to the weights in a way that the constraints will be again satisfied. We recall that whenever the constraint's value exceeds its boundary, variations corresponding to λ value are observed in the error signal.

Net. No.	α	σ	λ	$\mathbf{x}^*$	$F(x*)$
Ex. 3	**0.5**	**17.2**	**0.02**	**(4.44, −3.44)**	**−7.42**
[37]	–	–	–	(−0.45, 1.45)	−1.65

Tab. IV *The algorithm parameters for the objective function in Ex. 3 and the results in comparison to [37].*

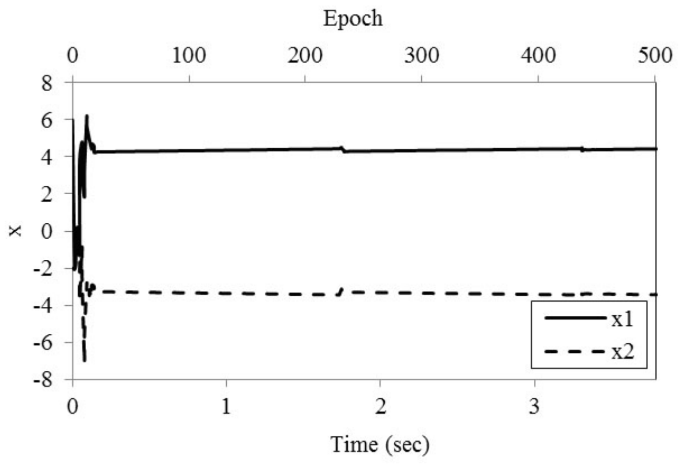

Fig. 9 *Transient behavior of the independent variables presented in Example 3.*

Example 4: Non-convex non-linear programming [38]; the multi-dimensional Rastrigin problem:

$$\begin{array}{ll} \min & f(\mathbf{x}) = A\,.n + \sum_{i=1}^{n}\left[x_i^2 - A\,.\cos(w\,.\,x_i\right] \\ \text{subject to} & \begin{cases} A = 10; n = 1:10; w = 2\pi \\ -10 < x_i < 10, i = 1,\dots,n \end{cases} \end{array} \tag{35}$$

is one of the test functions with a lot of local minima. Thereupon, the probability of trapping in the local minima is high, so that it may become difficult to find the global minimum. The two-dimensional Rastrigin function in the closed interval $[-5,5]$ is depicted in Fig. 10.

In this example, as well as the previous ones, the proposed algorithm has explored the solution space and found the optimal value of the problem correctly. For all of the decision variables the start point is 10. Fig. 11 confirms the convergence of the solution to the optimal value of zero in the case of ten-dimensional Rastrigin. For the n-dimensional Rastrigin function, $[n, 3, 4, 1 + 2n]$ is the structure of the network, and in all of the cases the parameters for the objective function are selected as: $\alpha = 0.5$, $\sigma = 10$, $\lambda = 0.05$, $\rho = 1$, $\gamma = 2$, and $C = 3$.

Example 5: Multi-objective non-convex non-linear programming [39]; the two-dimensional problem is declared as:

$$\begin{array}{ll} \max & \begin{cases} f_1(\mathbf{x}) = 3(1-x_1)^2\exp(-x_1^2-(x_2+1)^2) - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) \\ \qquad \exp(-x_1^2-x_2^2) - 3\exp(-(x_1+2)^2 - x_2^2) + \frac{1}{2}(2x_1+x_2) \\ f_2(\mathbf{x}) = 3(1+x_2)^2\exp(-x_2^2-(1-x_1)^2) - 10\left(-\frac{x_2}{5} + x_2^3 + x_1^5\right) \\ \qquad \exp(-x_2^2-x_1^2) - 3\exp(-(2-x_2)^2 - x_1^2) \end{cases} \\ \text{subject to} & -3 \le x_i \le 3, i = 1,2. \end{array} \tag{36}$$

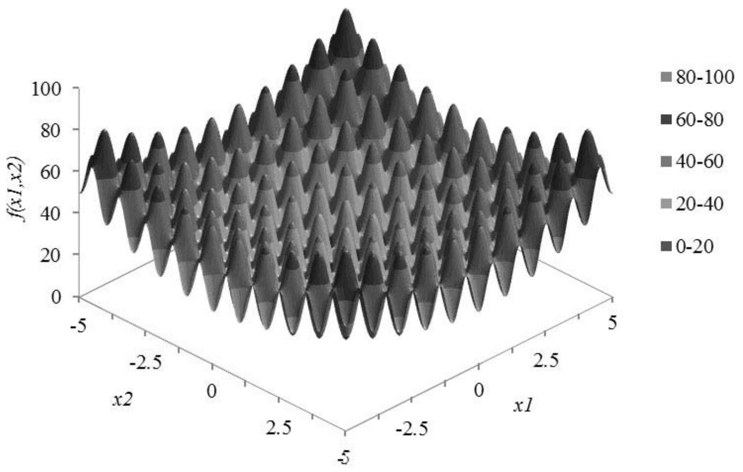

Fig. 10 *The two-dimensional Rastrigin function of the Example 4.*

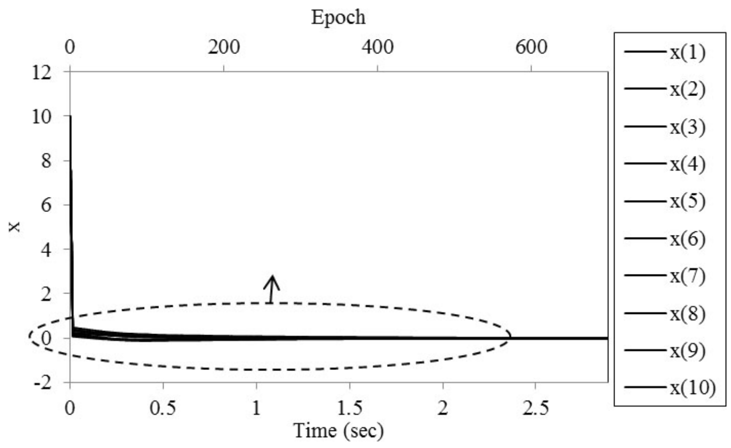

Fig. 11 *Transient behavior of the independent variables presented in Example 4.*

The Pareto front is exhibited in Fig. 12. With the initial point $x_0 = (1, 1)$, the solution reached by the proposed algorithm is $\mathbf{x}^* = (0.984, 0.548)$, as pointed out in Fig. 13, which is one of the non-dominated solutions discovered by Kim and De Weck [39]. The algorithm parameters with the structure of $[2, 3, 4, 6]$ for the objective functions are given as $\alpha = 0.5$, $\sigma = 8$, $\lambda = 0.05$, $\rho = 10$, $\gamma = 2$, and $C = 1$. For the constraints we choose: $\alpha = 0.5$, $\sigma = 2$, and $\lambda = 0.07$.

Example 6: Multi-objective non-convex non-linear programming [34,40]; the problem contains polynomials of the order 5 with 28 constraints, which embody 10 independent variables:

$$\min \begin{cases} f_1(\mathbf{x}) = 7x_1^2 - x_2^2 + x_1x_2 - 14x_1 - 16x_2 + 8(x_3-10)^2 \\ \qquad +4(x_4-5)^2 + (x_5-3)^2 + 2(x_6-1)^2 + 5x_7^2 \\ \qquad +7(x_8-11)^2 + 2(x_9-11)^2 + x_{10}^2 + 45 \\ f_2(\mathbf{x}) = (x_1-5)^2 + 5(x_2-12)^2 + 0.5x_3^4 + 3(x_4-11)^2 \\ \qquad +0.2x_5^5 + 7x_6^2 + 0.1x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \\ \qquad +x_8^2 + 3(x_9-5)^2 + (x_{10}-5)^2 \\ f_3(\mathbf{x}) = x_1^3 + (x_2-5)^2 + 3(x_3-9)^2 - 12x_3 + 2x_4^3 \\ \qquad +4x_5^2 + (x_6-5)^2 + 6x_7^2 + 3(x_7-2)x_8^2 - x_9x_{10} \\ \qquad +4x_9^3 + 5x_1 - 8x_1x_7 \end{cases} \tag{37}$$

$$\text{subject to} \begin{cases} g_1(\mathbf{x}) = -3(x_1-2)^2 - 4(x_2-3)^2 - 2x_3^2 + 7x_4 - 2x_5x_6x_8 + 120 \geq 0 \\ g_2(\mathbf{x}) = -5x_1^2 - 8x_2 - (x_3-6)^2 + 2x_4 + 40 \geq 0 \\ g_3(\mathbf{x}) = -x_1^2 - 2(x_2-2)^2 + 2x_1x_2 - 14x_5 - 6x_5x_6 \geq 0 \\ g_4(\mathbf{x}) = -0.5(x_1-8)^2 - 2(x_2-4)^2 - 3x_5^2 + x_5x_8 + 30 \geq 0 \\ g_5(\mathbf{x}) = 3x_1 - 6x_2 - 12(x_9-8)^2 + 7x_{10} \geq 0 \\ g_6(\mathbf{x}) = -4x_1 - 5x_2 + 3x_7 - 9x_8 + 105 \geq 0 \\ g_7(\mathbf{x}) = -10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0 \\ g_8(\mathbf{x}) = +8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0 \\ -5 \leq x_i \leq 15, i = 1, \ldots, 10. \end{cases}$$

Since the problem contains polynomials of high orders, it is prone to instability on account of its inherent divergence feature. Thus, it is crucial to select small initial weights. All the initial weights and decision variables are selected to be 0.01 and 10, respectively. The algorithm parameters are listed in Tab. V to gain insight into the parameters selection criteria for this kind of problems. Fig. 14 demonstrates the convergence of the independent variables to $\mathbf{x}^* = (0.4482, 4.1492, 4.6255, 4.1468, -0.1506, 4.0587, -0.6307, 8.9685, 5.6214, 13.0957)$.

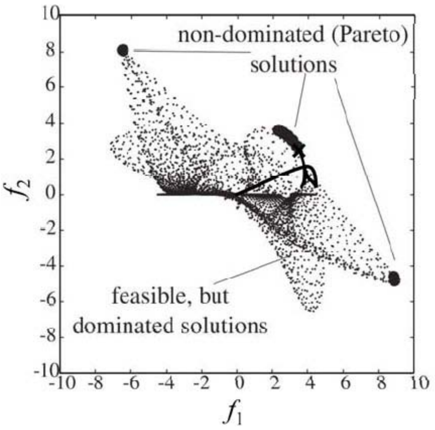

Fig. 12 *The Pareto front of the Example 5 offered in [39], where the exploration paths for the two functions to reach the front, i.e. non-dominated solutions, are depicted for the aim of comparison. The distinguished cross sign is the last point that the proposed algorithm has converged to.*

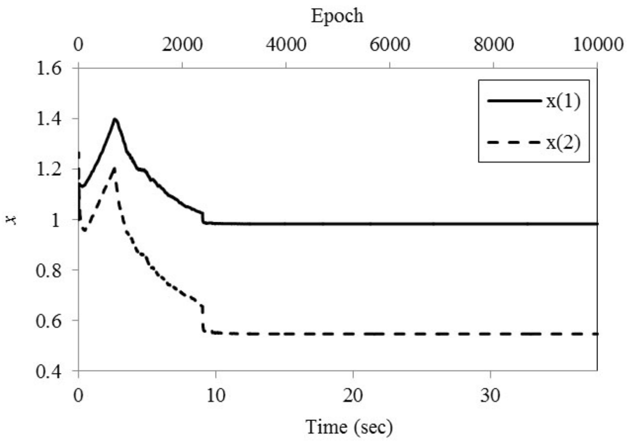

Fig. 13 *Convergence of the proposed algorithm to the optimal solution of the Example 5.*

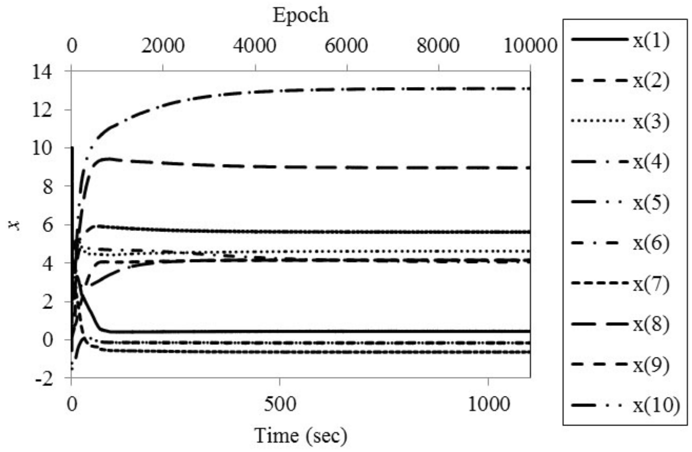

Fig. 14 *Transient response of the independent variables presented in Example 6.*

Net. No.	α	σ	λ	$\mathbf{x}^*$	$\mathbf{F}(\mathbf{x}^*)$
Ex. 6	**(0.5, 0.5, 0.5)**	**(500, 500, 500)**	**(0.01, 0.01, 0.01)**	**(0.4482, 4.1492, 4.6255, 4.1468, -0.1506, 4.0587, -0.6307, 8.9685, 5.6214, 13.0957)**	**(481,936,155)**
[34]	-	-	-	(1.9421, 2.8821, 5.3164, 5.6807, 0.2245, 1.2515, 1.1823, 6.1321, 6.1178, 7.7145)	(461,941,1146)
[40]	-	-	-	-	(536,611,582)

Tab. V *The algorithm parameters for the objective functions in Ex. 6 and the results in comparison to [34] and [40].*

The NN structure of $[10, 10, 10, 31]$ and the algorithm parameters $\rho = 80$, $\gamma = 1000$, and $C = 4$ are selected. For the constraints the parameters are set as $\alpha = 0.5$, $\sigma = 100$, and $\lambda = (0.01, 0.03, 0.03, 0.01, 0.03, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02)$.

Example 7: Multi-objective non-convex non-linear programming [41]; One of the problems frequently expressed in the multi-objective optimization is the Kursawe's study (KUR), which is a non-convex problem stated as:

$$\begin{aligned} \min \quad & \begin{cases} f_1(\mathbf{x}) = \sum\limits_{i=1}^{n-1} (-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})) \\ f_2(\mathbf{x}) = \sum\limits_{i=1}^{n} (|x_i|^{0.8} + 5\sin x_i^3) \end{cases} \\ \text{subject to} \quad & n = 3, -5 \leq x_i \leq 5, i = 1, 2, 3. \end{aligned} \tag{38}$$

This problem has two objectives between which there always exists a trade-off. Fig. 15 shows the Pareto front for this problem [41]. The spectral line on Fig. 15 points out the explored path by the proposed algorithm. As it is observed in Fig. 16, starting from the initial point $\mathbf{x}_0 = (10, 10, 10)$, the algorithm will eventually converge to the point specified with the cross sign in the location $(f_1(\mathbf{x}^*), f_2(\mathbf{x}^*)) = (-17.96, -3.87)$ corresponding to $\mathbf{x}^* = (-1.14, 0.0, 0.0)$. With the network structure of $[3, 5, 5, 8]$, the parameters adjusted for the objective functions are $\alpha = (0.5, 0.5)$, $\sigma = (10, 10)$, and $\lambda = (0.05, 0.05)$, and for the constraints are $\alpha = 0.5$, $\sigma = 3$, and $\lambda = 0.07$. Algorithm parameters are set to be $\rho = 1$, $\gamma = 2$, and $C = 1$.

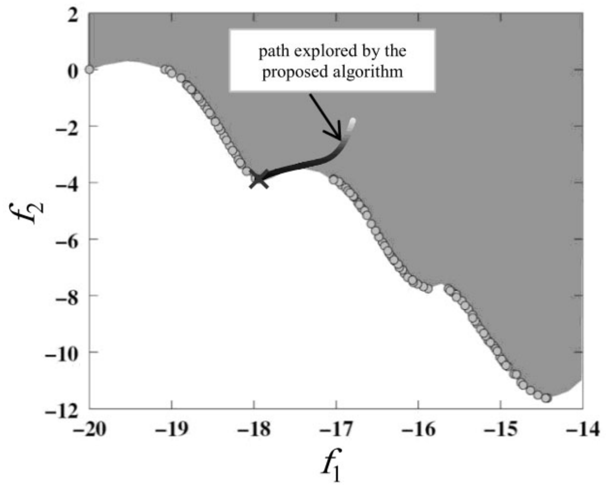

Fig. 15 *The Pareto front of Example 7 used in [41], where the exploration path for the two functions to reach the front is shown for the aim of comparison. The specified cross sign is the last point to which the proposed algorithm converges.*

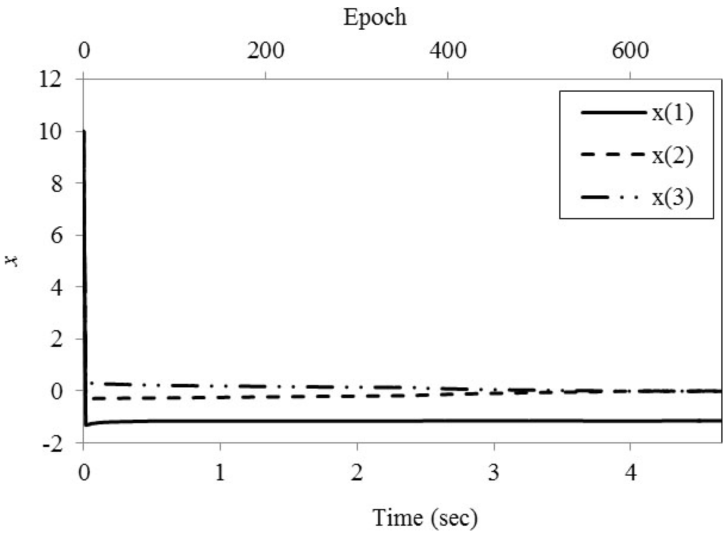

Fig. 16 *Transient performance of the decision variables declared in Example 7.*

6. Conclusion

A new method based on considering the objective function as the activation function of a neuron in the output layer of a multi-dimensional MLP neural network was introduced and expanded for the multi-objective non-linear programming. During the training process, by minimizing the error of each objective function and consequently the network energy, algorithm reaches the optimal value of the functions. Network training algorithm and the effective parameters in the training procedure all directly affect the speed and the potentiality of the global search. Hence, a modified dynamic learning rate was utilized to avoid the local minima. Regarding an appropriate definition for the error signal, the proposed method can be implemented in order to optimize objective functions or to solve them with respect to predefined objectives. The constraints treated in the same manner as the objective functions in the sense that a proper error signal would be defined for them. Various examples in the field of multi-objective non-convex non-linear programming were evaluated. In view of the results, the high speed and the capability of the proposed method in global searching of the solution space were verified. In addition, it can be deduced that, there is no need for fundamental changes in the structure of the network for different problems of concern.

References

[1] C. Grosan, A. Abraham: On a class of global optimization test functions, Neural Network World, vol. 19, no. 2, pp. 247-252, 2009.

[2] M. A. Valdebenito and G. I. Schuëller: A survey on approaches for reliability-based optimization, Structural and Multidisciplinary Optimization, vol. 42, no. 5, pp. 645-663, 2010.

[3] L. K. Li, S. S. Shao: A neural network approach for global optimization with applications, Neural Network World, vol. 3, no. 10, pp. 491-508, 2008.

[4] U. Wen, K. Lan, H. Shih: A review of Hopfield neural networks for solving mathematical programming problems, European Journal of Operational Research, vol. 198, no. 3, pp. 675-687, 2009.

[5] A. Hosseini, J. Wang, S. M. Hosseini: A recurrent neural network for solving a class of generalized convex optimization problems, Neural Networks, vol. 44, pp. 78-86, 2013.

[6] E. Irigoyen, M. Larrea, J. Valera, V. Gómez, F. Artaza: A hybridized neuro-genetic solution for controlling industrial R3 workspace, Neural Network World, vol. 20, no. 7, pp. 811-824, 2010.

[7] A. Zhou, B. Y. Qu, H. Li, S. Z. Zhao, P. N. Suganthan, Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," Swarm and Evolutionary Computation, vol. 1, no. 1, pp. 32-49, 2011.

[8] A. Ponsich, A. L. Jaimes, C. A. C. Coello: A survey on multi-objective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications, IEEE Transactions on Evolutionary Computation, vol. 17, no. 3, pp. 321-344, 2012.

[9] B. A. Conway: A survey of methods available for the numerical optimization of continuous dynamic systems, Journal of Optimization Theory and Applications, vol. 152, no. 2, pp. 271-306, 2012.

[10] S. A. Marhon, C. J. Cameron, S. C. Kremer: Recurrent Neural Networks, Handbook on Neural Information Processing, Springer, pp. 29-65, 2013.

[11] S. Jung, S. S. Kim: Hardware implementation of a real-time neural network controller with a DSP and an FPGA for nonlinear systems, IEEE Transactions on Industrial Electronics, vol. 54, no. 1, pp. 265-271, 2007.

[12] J. Reifman, E. E. Feldman: Multilayer perceptron for nonlinear programming, Computers and Operations Research, vol. 29, no. 9, pp. 1237-1250, 2002.

[13] R. Francelin, I. Ricarte, F. Gomide: System optimization with artificial neural networks: parallel implementation using transputers, in IEEE International Joint Conference on Neural Networks, pp. 630-635, 1992.

[14] X. Yang, K. Tamura, T. Shen: An approach to solve nonlinear $H\infty$ control problem based on neural networks, in IEEE Proceedings of the 34th SICE Annual Conference, pp. 1245-1249, 1995.

[15] K. Hornik: Approximation capabilities of multilayer feedforward networks, Neural Networks, vol. 4, no. 2, pp. 251-257, 1991.

[16] J. Moody, N. Yarvin: Networks with learned unit response functions, Advances in Neural Information Processing Systems, pp. 1048-1048, 1993.

[17] K. Halawa: A method to improve the performance of multilayer perceptron by utilizing various activation functions in the last hidden layer and the least squares method, Neural Processing Letters, vol. 34, no. 3, pp. 293-303, 2011.

[18] T. Ebert, O. Bänfer, O. Nelles: Multilayer perceptron network with modified sigmoid activation functions, Artificial Intelligence and Computational Intelligence, Springer, pp. 414-421, 2010.

[19] M. Ö. Efe: Novel neuronal activation functions for feedforward neural networks, Neural Processing Letters, vol. 28, no. 2, pp. 63-79, 2008.

[20] M. Solazzi, A. Uncini: Regularising neural networks using flexible multivariate activation function, Neural Networks, vol. 17, no. 2, pp. 247-260, 2004.

[21] M. Sartori, M. Reggiani, A. J. van den Bogert, D. G. Lloyd: Estimation of musculotendon kinematics in large musculoskeletal models using multidimensional B-splines, Journal of Biomechanics, vol. 45, no. 3, pp. 595-601, 2012.

[22] J. De Andrés, P. Lorca, F. J. de Cos Juez, F. Sánchez-Lasheras: Bankruptcy forecasting: A hybrid approach using fuzzy c-means clustering and multivariate adaptive regression splines (MARS), Expert Systems with Applications, vol. 38, no. 3, pp. 1866-1875, 2011.

[23] P. J. García Nieto, J. Martínez Torres, F. J. de Cos Juez, F. Sánchez Lasheras: Using multivariate adaptive regression splines and multilayer perceptron networks to evaluate paper manufactured using Eucalyptus globulus, Applied Mathematics and Computation, vol. 219, no. 2, pp. 755-763, 2012.

[24] I. Isa, S. Omar, Z. Saad, M. Osman: Performance comparison of different multilayer perceptron network activation functions in automated weather classification, in Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, pp. 71-75, 2010.

[25] I. Isa, Z. Saad, S. Omar, M. Osman, K. Ahmad, H. M. Sakim: Suitable MLP network activation functions for breast cancer and thyroid disease detection, in Second International Conference on Computational Intelligence, Modelling and Simulation, pp. 39-44, 2010.

[26] S. S. Haykin, Neural networks: a comprehensive foundation: Prentice Hall Englewood Cliffs, NJ, 2008.

[27] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuisen, Evolutionary algorithms for solving multi-objective problems: Springer, 2007.

[28] R. Zhang, Z. B. Xu, G. B. Huang, D. Wang: Global convergence of online BP training with dynamic learning rate, IEEE Transactions on Neural Networks and Learning Systems, vol. 23, no. 2, pp. 330-341, 2012.

[29] L. Behera, S. Kumar, A. Patnaik: On adaptive learning rate that guarantees convergence in feedforward networks, IEEE Transactions on Neural Networks, vol. 17, no. 5, pp. 1116-1125, 2006.

[30] A. S. Shafie, I. A. Mohtar, S. Masrom, N. Ahmad: Backpropagation neural network with new improved error function and activation function for classification problem, in IEEE Symposium on Humanities, Science and Engineering Research, pp. 1359-1364, 2012.

[31] L. K. Li, R. C. H. Lee: New error function for single hidden layer feedforward neural networks, in Congress on Image and Signal Processing, pp. 752-755, 2008.

[32] J. Pastor-Pellicer, F. Zamora-Martínez, S. Espaa-Boquera, M. J. Castro-Bleda: F-measure as the error function to train neural networks, Advances in Computational Intelligence, pp. 376-384: Springer, 2013.

[33] M.-P. Jarabo-Amores, M. Rosa-Zurera, R. Gil-Pita, F. López-Ferreras: Study of two error functions to approximate the Neyman–Pearson detector using supervised learning machines, IEEE Transactions on Signal Processing, vol. 57, no. 11, pp. 4175-4181, 2009.

[34] L. L. Schumaker: Spline functions: basic theory, Cambridge Univ, 2007.

[35] Y. Xia, J. Wang: A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints, IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 7, pp. 1385-1394, 2004.

[36] M. A. Abo-Sinna, A. H. Amer, "Extensions of TOPSIS for multi-objective large-scale nonlinear programming problems," Applied Mathematics and Computation, vol. 162, no. 1, pp. 243-256, 2005.

[37] W. Bian, X. Xue: Subgradient-based neural networks for nonsmooth nonconvex optimization problems, IEEE Transactions on Neural Networks, vol. 20, no. 6, pp. 1024-1038, 2009.

[38] O. Montiel, O. Castillo, P. Melin, A. R. Díaz, R. Sepúlveda, "Human evolutionary model: A new approach to optimization," Information Sciences, vol. 177, no. 10, pp. 2075-2098, 2007.

[39] I. Y. Kim, O. De Weck: Adaptive weighted-sum method for bi-objective optimization: Pareto front generation, Structural and Multidisciplinary Optimization, vol. 29, no. 2, pp. 149-158, 2005.

[40] M. Sakawa, K. Yauchi: An interactive fuzzy satisficing method for multiobjective nonconvex programming problems with fuzzy numbers through coevolutionary genetic algorithms, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 31, no. 3, pp. 459-467, 2001.

[41] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan: A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, 2002.