An Evolutionary Multi-layer Perceptron Neural Network for Solving Unconstrained Global Optimization Problems

Jui-Yu Wu*
Department of Business Administration
Lunghwa University of Science and Technology, Taoyuan, Taiwan
jywu@mail.lhu.edu.tw

***Abstract*—This study presents an evolutionary multi-layer perceptron neural network (EvoMLPNN) method, which consists of an MLPNN and an improved quantum-behaved particle swarm optimization (IQPSO) method. This study develops a network topology of an MLPNN that can be used to solve unconstrained global optimization (UGO) problems, and optimizes the weights of the MLPNN by using the IQPSO approach. To evaluate the performance of the proposed EvoMLPNN approach, a set of benchmark UGO problems was used and the numerical results obtained using the EvoMLPNN method were compared with those obtained using published algorithms. Experimental results show that the proposed EvoMLPNN method can find a global optimization solution for each test UGO problem and can solve highly dimensional UGO problems, and that the numerical results of the EvoMLPNN approach outperform to those of some published algorithms.**

Keywords—multi-layer perceptron; neural networks; quantum-behaved particle swarm optimization; unconstrained global optimization

I. INTRODUCTION

A multi-layer perceptron neural network (MLPNN) is a feedforward NN and has many advantages, such as parsimoniousness and universal approximation [1]. Many MLPNNs have been successfully used to various applications. For instance, Kolay and Baser [2] used an MLPNN and a general linear model (GLM) to estimate the dry unit weight of different types of soils, and indicated that an MLPNN can obtain better dry unit weight estimates than a GLM method. Mabu et al., [3] proposed a rule-based evolutionary algorithm (EA) using an MLPNN to enhance the performance of the stock trading system.

An MLPNN updates weights of a network topology by using a supervised learning algorithm that is the error back propagation (EBP) method based on a gradient information of the minimizing an objective function (energy function), namely a back-propagation NN (BPNN). The learning algorithm of the EBP method has a limitation that is easily to find local optima. To overcome this drawback, a scaled conjugate gradient algorithm (SCGA) [4] has been developed. Although the SCGA can guarantee to obtain the global optima, the method still requires a gradient information based on an energy function. Fortunately, many stochastic global optimization (SGO) methods that consist of random search techniques, EAs, swarm intelligence (SI) and other approaches (such as harmony search, memetic algorithms, cultural algorithms, scatter search and tunneling methods) [5] have been developed. Several EA and SI approaches have been used to optimize the weights of a network topology for an MLPNN. For instances, Ludermir and de Oliveira [6] optimized the weights and architectures of an MLPNN by using particle swarm optimization (PSO) methods and used the MLPNN to identified the factors related to common mental disorder. Liu et al., [7] found the weights of an MLPNN by using a genetic algorithm with a PSO method and employed the MLPNNs to forecast non-stationary wind speeds. Moreover, the implementation of these SGO methods often must be predefined some parameter settings.

A quantum-behaved PSO (QPSO) method has several advantages, such as (1) outperforms to a standard PSO method, (2) requires less parameter settings and (3) is a global convergence guaranteed optimizer [8-10]. Various QPSO methods have been used to many applications. For instance, Sun et al., [11] used an enhanced QPSO approach for solving constrained optimization problems. Li et al., [12] developed a dynamic-context cooperative QPSO algorithm to optimize the parameters of Otsu image segmentation for processing medical images. Turgut [13] employed a hybrid chaotic QPSO algorithm for thermal design of plate fin heat exchangers.

An unconstrained global optimization (UGO) problem can be defined as follows:

$$\text{Minimize } f(\mathbf{x}), \quad \mathbf{x}=[x_1,x_2,\ldots,x_N]^T \in \Re^N \tag{1}$$

where $f:\Re^N \to \Re$ is an real-valued objective function and $\mathbf{x} \in \Omega$, represents a decision variable vector. Additionally, $\Omega \subseteq \Re^N$ denotes search space (Ω), which is *N*-dimensional and bounded by parametric constraints, as follows:

$$x_n^l \le x_n \le x_n^u, \quad n=1,2,\ldots,N \tag{2}$$

where x_n^l and x_n^u are the lower and upper boundaries of the decision variables x_n. Various SGO methods have been applied to solve UGO problems [8, 14, 15].

A network topology of an MLPNN can be easily to extend based on a solving problem. Therefore, this study develops a network topology of an MLPNN that can solve UGO problems and optimizes weights of the network topology by using an improved QPSO (IQPSO) method. The proposed approach is called an evolutionary MLPNN (EvoMLPNN). The performance of the EvoMLPNN approach is evaluated by

978-1-5090-0806-3/16/$31.00 copyright 2016 IEEE
ICIS 2016, June 26-29, 2016, Okayama, Japan

using a set of benchmark UGO problems and the numerical results of the proposed EvoMLPNN are compared with those of published algorithms.

II. Related work

A. Multi-layer perceptron neural network (MLPNN)

A network topology of an MLPNN consists of one input layer, at least one hidden layer of neurons (computational nodes) and one output layer of neurons, as shown in Fig.1.

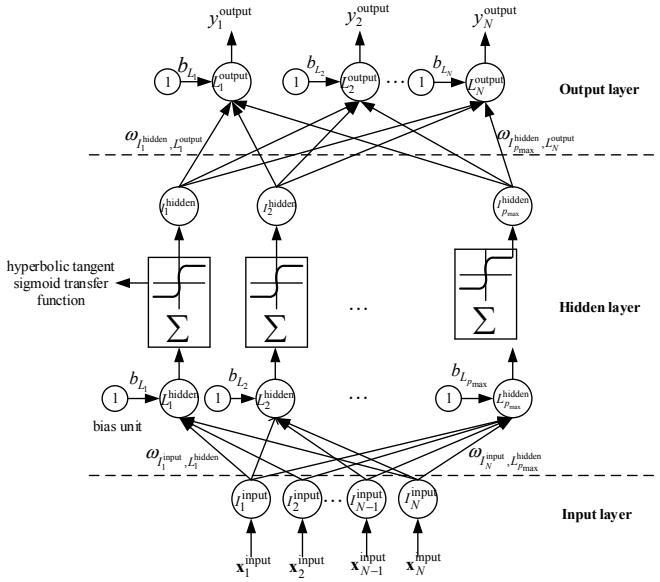

Fig. 1. A network topology of an MLPNN

The input layer collects input patterns. The hidden layer sums input and bias weights and then generates corresponding outputs by using a hyperbolic tangent sigmoid activation function. The output layer creates final responses. An MLPNN with single hidden layer has been proved that an MLPNN can has a universal approximation capability for any complex nonlinear function [2, 16-18]. The notations in Fig.1 are determined as follows:

$\mathbf{x}_1^{\text{input}}$ = input vector n ($n = 1, 2, ..., N$);

$\omega_{I_n^{\text{input}}, L_p^{\text{hidden}}}$ = weight between a neuron I_n^{input} ($n = 1, 2, ..., N$) in the input layer and a neuron L_p^{hidden} ($p = 1, 2, ..., p_{\max}$) in the hidden layer;

$p_{\max}$ = maximum number of neurons in the hidden layer;

b_{L_p} = bias weight between a bias unit b and a neuron L_p^{hidden} ($p = 1, 2, ..., p_{\max}$) in the hidden layer;

$\omega_{I_p^{\text{hidden}}, L_n^{\text{output}}}$ = weight between an input neuron I_p^{hidden} ($p = 1, 2, ..., p_{\max}$) in the hidden layer and a neuron L_n^{output} ($n = 1, 2, ..., N$) in the output layer;

b_{L_n} = bias weight between a bias unit b and a neuron L_n^{output} ($n = 1, 2, ..., N$) in the output layer;

y_n^{output} = network output of a neuron L_n^{output} ($n = 1, 2, ..., N$) in the output layer

B. Quantum-behaved PSO Method

A pseudo-code of a QPSO approach [19] is shown in Fig.2, which is described as follows:

Procedure QPSO Method
begin
$g \leftarrow 0$
Step 1: initialize parameter settings
while $g < g_{\max}$ **do**
Step 2： calculate the position of the mean best particle
Step 3： update the positions of the personal particles
Step 4： obtain contraction expansion coefficient
Step 5： update the positions of the particles
Step 6： compute the fitness function value
Step 7： obtain the positions of the local and global best particles
$g \leftarrow g + 1$
endWhile
end

Fig. 2. A pseudo-code of a QPSO method

Step 1: initialize parameter settings

The parameter settings, such as population size (ps), maximum generation number ($g_{\max}$), initial contraction expansion coefficient ($\alpha(0)$) and the lower and upper boundaries of the candidate solutions ($x^l_{\text{QPSO},n}, x^u_{\text{QPSO},n}$) ($n = 1, 2, ..., N$) for the QPSO approach are determined. An initial population (particle swarm) is randomly generated based on a ps from [$x^l_{\text{QPSO},n}, x^u_{\text{QPSO},n}$] and then current position $x_{\text{QPSO},j,n}(0)$ ($j = 1, 2, ..., ps, n = 1, 2, ..., N$), initial local best position $p^{lb}_{j,n}(0)$ and initial global best position $p^{gb}_{j,n}(0)$ can be obtained.

Step 2: calculate the position of the mean best particle

The position of the mean best particle at generation g can be computed as follows:

$$C_n(g) = (C_1(g), C_2(g), \ldots, C_N(g)), n = 1, 2, \ldots, N = \left(\frac{1}{ps}\sum_{j=1}^{ps} p^{lb}_{j,1}(g), \frac{1}{ps}\sum_{j=1}^{ps} p^{lb}_{j,2}(g), \ldots, \frac{1}{ps}\sum_{j=1}^{ps} p^{lb}_{j,N}(g) \right) \quad (3)$$

Step 3: update the positions of the personal particles

The positions of the personal particles at generation g can be obtained by using Eq. (4), as follows:

$$p_{j,n}(g) = \varphi p^{lb}_{j,n}(g) + (1-\varphi) p^{gb}_{j,n}(g) \quad j = 1, 2, ..., ps, n = 1, 2, ..., N \quad (4)$$

where φ = uniform random number in the interval [0, 1].

Step 4: obtain contraction expansion coefficient

A parameter α can be adjusted by using Eq. (5) [20], as follows:

$$\alpha(g+1)=\alpha(0)+(g_{\max}-g)\left(\frac{(\alpha(g)-\alpha(0))}{g_{\max}}\right) \quad (5)$$

where

$\alpha(g)$ = contraction expansion coefficient at generation g

$\alpha(g+1)$ = contraction expansion coefficient at generation $g+1$

Step 5: update the positions of the particles

The positions of the particles at generation $g+1$ can be updated by using Eq. (6), as follows:

$$\begin{cases} x_{\mathrm{QPSO},j,n}(g+1)=p_{j,n}(g)+\alpha\left|C_n(g)-x_{\mathrm{QPSO},j,n}(g)\right|\ln\left(1/u_{j,n}(g+1)\right), U(0,1)\geq 0.5 \\ x_{\mathrm{QPSO},j,n}(g+1)=p_{j,n}(g)-\alpha\left|C_n(g)-x_{\mathrm{QPSO},j,n}(g)\right|\ln\left(1/u_{j,n}(g+1)\right), U(0,1)<0.5 \end{cases}$$
$$j=1,2,...,ps,\ n=1,2,...,N \quad (6)$$

where $U(0,1)$ = uniform random number in the interval [0, 1]. A new population of candidate solutions is thus created.

Step 6: compute the fitness function value

The fitness function value $f(\mathbf{x}_{\mathrm{QPSO},j})$ ($j=1,2,...,ps$) can be computed based on an objective function of a solving problem.

Step 7: obtain the positions of the local and global best particles

A pair-wise comparison is made between the value $f(\mathbf{x}_{\mathrm{QPSO},j})$ ($j=1,2,...,ps$) of candidate solutions in the new population and that in the current population, and then the solutions with the strong $f(\mathbf{x}_{\mathrm{QPSO},j})$ are remained. The $p_{j,n}^{lb}(g+1)$ thus can be created. If the fitness value of the $p_{j,n}^{lb}(g+1)$ outperforms to that of $p_{j,n}^{gb}(g)$, then the $p_{j,n}^{lb}(g+1)$ replaces the $p_{j,n}^{gb}(g)$.

Repeat steps 2-7 until the termination criterion $g_{\max}$ is met.

III. The Proposed EvoMLPNN Method

This study develops an EvoMLPNN approach, which optimizes the weights of a network topology of an MLPNN by using an IQPSO method, for solving UGO problems. The network topology of the proposed EvoMLPNN method is shown in Fig.3. The decision variables of an UGO problem is used as an input pattern for the MLPNN and are then normalized. These re-normalized output neurons of the MLPNN represent decision variables $x_{\mathrm{EvoMLPNN},n}$ ($n=1,2,...,N$) of a solving UGO problem and become as the input pattern of the MLPNN for next generation. The weights of the MLPNN represent candidate solutions (particles) for the IQPSO approach, as shown in Fig.4.

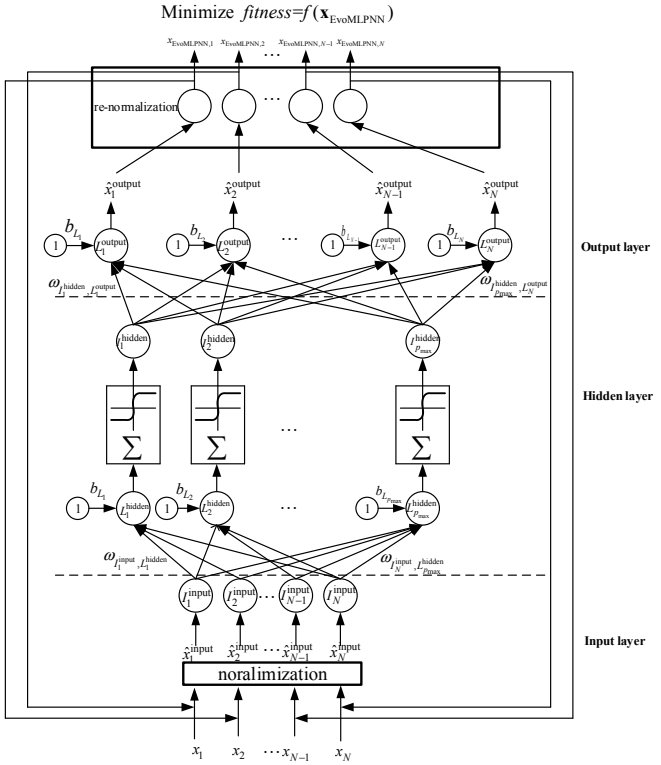

Fig. 3. The network topology of the proposed EvoMLPNN method

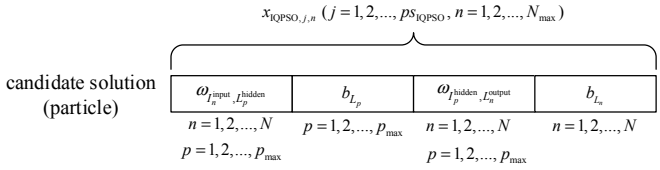

Fig. 4. The representation of a candidate solution for the QPSO method

Figure 5 shows the pseudo-code of the proposed EvoMLPNN approach, which is described as follows.

```
Procedure EvoMLPNN Method
begin
  g ← 0
  Step 1: initialize parameter settings and normalize initial decision variables
  while g < g_max,IQPSO do
    Step 2 : calculate the position of the mean best particle
    Step 3 : update the positions of the personal particles
    Step 4 : obtain contraction expansion coefficient
    Step 5 : update the positions of the particles
    Step 6 : implement a re-normalization process
    Step 7 : perform an elitist strategy
    g ← g + 1
  endWhile
end
```

Fig. 5. The pseudo-code of the proposed EvoMLPNN approach

Step 1: initialize parameter settings and normalize initial decision variables

Many parameter settings of the proposed EvoMLPNN are initialized, as follows:

[$\omega^{l}_{\text{input-hidden}}, \omega^{u}_{\text{input-hidden}}$] = upper and lower boundaries of weights in the input-hidden layers; [$\omega^{l}_{\text{hidden,bias}}, \omega^{u}_{\text{hidden,bias}}$] = upper and lower boundaries of weights of bias units in the hidden layer; [$\omega^{l}_{\text{hidden-output}}, \omega^{u}_{\text{hidden-output}}$] = upper and lower boundaries of weights in the hidden-output layers; [$\omega^{l}_{\text{output,bias}}, \omega^{u}_{\text{output,bias}}$] = upper and lower boundaries of weights of bias units in the output layers; $p_{\max}$; ps_{IQPSO} = population size of the IQPSO method; $\alpha(0)$; $g_{\max,\text{IQPSO}}$ = maximum generation number of the IQPSO method; $p_{m,\text{IQPSO}}$ = mutation probability of the IQPSO method.

Each decision variable (x_n) of an UGO problem undergoes a normalization process, the normalized decision variables become as input variables for the EvoMLPNN method. The normalization process is defined as follows:

$$\hat{x}_n^{\text{input}} = \frac{x_n^{\text{input}} - x_n^{l}}{x_n^{u} - x_n^{l}}(E_{\max} - E_{\min}) + E_{\min}, \quad n = 1, 2, ..., N \tag{7}$$

where

$\hat{x}_n^{\text{input}}$ = normalized input decision variable (x_n);

$E_{\min}$ = minimum value of expected output,

$E_{\max}$ = maximum value of expected output.

The values [$E_{\min}$, $E_{\max}$] are set to [0.2, 0.8]. Moreover, an initial population is created based on the ps_{IQPSO} from the [$\omega^{l}_{\text{input-hidden}}, \omega^{u}_{\text{input-hidden}}$], [$\omega^{l}_{\text{hidden,bias}}, \omega^{u}_{\text{hidden,bias}}$], [$\omega^{l}_{\text{hidden-output}}, \omega^{u}_{\text{hidden-output}}$] and [$\omega^{l}_{\text{output,bias}}, \omega^{u}_{\text{output,bias}}$]. A candidate solution (particle) consists of weights. A size of a candidate solution can be computed as follows:

$$N_{\max} = 2 \times (N \times p_{\max}) + p_{\max} + N \tag{8}$$

For instance, an UGO problem with 30 decision variables and $p_{\max} = 2$, the size of candidate solution (particle) equals "152".

Step 2: calculate the position of the mean best particle

According to Eq. (3), the $C_n(g)$ ($n = 1, 2, ..., N_{\max}$) can be obtained.

Step 3: update the positions of the personal particles

Related to Eq. (4), the $p_{j,n}(g)$ ($n = 1, 2, ..., N_{\max}$, $j = 1, 2, ..., ps_{\text{IQPSO}}$) can be found.

Step 4: obtain contraction expansion coefficient

This study develops a method to update the contraction expansion coefficient, as follows:

$$\alpha(g+1) = \alpha(0) + U_1(0,1) \times \left(\frac{g}{g_{\max,\text{IQPSO}}} \right)^2 \tag{9}$$

where $\alpha(0)$=0.01 and $U_1(0,1)$ = uniform random number in the interval [0, 1].

Step 5: update the positions of the particles

This study uses a mutation operator based on a Cauchy distribution [21]. The location and scale parameters of Cauchy distribution are $p_{j,n}(g)$ and $\left(C_n(g) - p_{j,n}(g)\right)$ ($j = 1, 2, ..., ps_{\text{IQPSO}}$, $n = 1, 2, ..., N_{\max}$), respectively. The perturbed factor is defined as follows:

$$\theta_{j,n}(g) = C_{\text{mutation}}(p_{j,n}(g), C_n(g) - p_{j,n}(g)) \quad j = 1, 2, ..., ps_{\text{IQPSO}}, n = 1, 2, ..., N_{\max} \tag{10}$$

An uniform random number $U_2(0,1)$ is created. When $U_2(0,1) > 0.5$, Eq. (11) is used; otherwise, Eq. (12) is employed.

$$\begin{cases} x_{\text{IQPSO},j,n}(g+1) = p_{j,n}(g) + \alpha \left| C_n(g) - x_{\text{IQPSO}j,n}(g) \right| \ln\left(1/u_{j,n}(g+1)\right), U_3(0,1) \geq 0.5 \\ x_{\text{IQPSO}j,n}(g+1) = p_{j,n}(g) - \alpha \left| C_n(g) - x_{\text{IQPSO}j,n}(g) \right| \ln\left(1/u_{j,n}(g+1)\right), U_3(0,1) < 0.5 \end{cases}$$
$$j = 1, 2, ..., ps_{\text{IQPSO}}, n = 1, 2, ..., N_{\max} \tag{11}$$

$$\begin{cases} x_{\text{IQPSO}j,n}(g+1) = \theta_{j,n}(g) + \alpha(g) \left| p_{j,n}(g) - x_{\text{IQPSO}j,n}(g) \right| \ln\left(1/u_{j,n}(g+1)\right), U_4(0,1) \geq 0.5 \\ x_{\text{IQPSO}j,n}(g+1) = \theta_{j,n}(g) - \alpha(g) \left| p_{j,n}(g) - x_{\text{IQPSO}j,n}(g) \right| \ln\left(1/u_{j,n}(g+1)\right), U_4(0,1) < 0.5 \end{cases}$$
$$j = 1, 2, ..., ps_{\text{IQPSO}}, n = 1, 2, ..., N_{\max} \tag{12}$$

where $U_3(0,1)$ and $U_4(0,1)$ = uniform random number in the interval [0, 1].

The candidate solutions are limited in the interval [$\omega^{l}_{\text{input-hidden}}, \omega^{u}_{\text{input-hidden}}$], [$\omega^{l}_{\text{hidden,bias}}, \omega^{u}_{\text{hidden,bias}}$], [$\omega^{l}_{\text{hidden-output}}, \omega^{u}_{\text{hidden-output}}$] and [$\omega^{l}_{\text{output,bias}}, \omega^{u}_{\text{output,bias}}$].

Step 6: implement a re-normalization process

Each value of output neuron undergoes a re-normalization process to create each candidate solution of an UGO problem. If a created candidate solution $x_{\text{EvoMLPNN},n}$ ($n = 1, 2, ..., N$) exceeds its x_n^{u} ($n = 1, 2, ..., N$), then the candidate solution $x_{\text{EvoMLPNN},n}$ equals the x_n^{u} ; if a generated candidate solution $x_{\text{EvoMLPNN},n}$ is smaller than its x_n^{l} ($n = 1, 2, ..., N$), then the candidate solution $x_{\text{EvoMLPNN},n}$ is equivalent to x_n^{l} .

Step 7: perform an elitist strategy

A value of an objective function of an UGO problem is used as a fitness function value, as follows:

$$\text{Minimize } f(\mathbf{x}_j) = fitness_j = f(\mathbf{x}_{\text{EvoMLPNN},j}) \quad j = 1, 2, ..., ps_{\text{IQPSO}} \tag{12}$$

A fitness function value of the created candidate solution can be evaluated. A pair-wise comparison is made between the

fitness function value $f(\mathbf{x}_{\text{EvoMLPNN},j})$ ($j = 1, 2, ..., ps_{\text{IQPSO}}$) of candidate solutions in the new population and that in the current population, and then the solutions with the strong $f(\mathbf{x}_{\text{EvoMLPNN},j})$ are remained. The $p_{j,n}^{lb}(g+1)$ ($j = 1, 2, ..., ps_{\text{IQPSO}}$, $n = 1, 2, ..., N_{\max}$,) thus can be obtained. The solution with the best fitness function value is selected as the $p_{j,n}^{lb}(g+1)$ ($j = 1, 2, ..., ps_{\text{IQPSO}}$, $n = 1, 2, ..., N_{\max}$,) and is used as a next input pattern of the MLPNN. The elitist strategy guarantees that the best candidate solution is always preserved in the next generation. The current particle swarm is updated to the particle swarm of the next generation.

Steps 2 and 7 are repeated until the termination condition is satisfied.

IV. Results

The proposed EvoMLPNN approach was coded in MATLAB software and run on an Intel Core i3 Pentium Dual-Core Processor 3.5 (GHz) personal computer. A set of test problems (TPs) were collected from several published literatures [14, 22-25]. One-hundred independent runs were conducted for each TP. Numerical results were summarized, including rate of successful minimizations (success rate %), best, mean, worst, mean computational CPU time (MCCT) and mean error *ME* (average value of the gap between the objective function values calculated using the EvoMLPNN solution and the known global minimum value). The parameter settings for the proposed EvoMLPNN method is as follows:

$[\omega_{\text{input-hidden}}^{l}, \omega_{\text{input-hidden}}^{u}] = [0.0001, 0.5]$;

$[\omega_{\text{hidden-output}}^{l}, \omega_{\text{hidden-output}}^{u}] = [0.0001, 0.5]$;

$[\omega_{\text{hidden,bias}}^{l}, \omega_{\text{hidden,bias}}^{u}] = [0, 0.1]$; $[\omega_{\text{output,bias}}^{l}, \omega_{\text{output,bias}}^{u}] = [0, 0.1]$;

$ps_{\text{IQPSO}} = 100$; $\alpha(0) = 0.01$; $p_{m,\text{IQPSO}} = 0.5$, and $p_{\max} = 2$.

Table I lists the numerical results obtained using the EvoMLPNN approach and shows that the proposed method can find the global solution for each TP, and that the proposed method can be applied to an UGO problem with high-dimensional space (N = 50 and 100). Table II compares the numerical results obtained using the EvoMLPNN method with those obtained using published algorithms and indicates that the numerical results of the EvoMLPNN method outperform to those of NM-PSO [25], PSACO [23], GA-PSO [25], DE-PSO [26], AM-PSO1 [26], AMPSO2 [26], CHA [23] and CTSS [23] methods for TPs 2 and 3, and that the numerical results obtained using the EvoMLPNN method is better than those obtained using PSACO [23], GA-PSO [25], DE-PSO [26], AM-PSO1 [26], AMPSO2 [26], CHA [23] and RGA-PSO [22] for TP 4. Moreover, the numerical results obtained using the EvoMLPNN approach is superior to those obtained using AIA-PSO [22] and RGA-PSO [22] methods for TP 6.

V. Conclusion

This study develops an EvoMLPNN approach that integrates an IQPSO approach and an MLPNN. The proposed EvoMLPNN method was applied to a set of benchmark UGO problems. Numerical results show that the proposed method can obtain a global solution for each test UGO problem with a high-dimensional space (N = 50 and 100), and that the numerical results obtained using the proposed EvoMLPNN method outperform to those obtained published algorithms for test UGO problems. Future work will apply the proposed EvoMLPNN method to CEC 2005 and CEC 2004 unconstrained benchmark functions.

TABLE I. Numerical results obtained using the proposed EvoMLPNN approach

TP No.	Function name	global minimum	required accuracy	$g_{\max,\text{IQPSO}}$	success rate (%)	$f(\mathbf{x}^{*}_{\text{EvoMLPNN}})$	$f(\mathbf{x}^{\text{mean}}_{\text{EvoMLPNN}})$	$f(\mathbf{x}^{\text{worst}}_{\text{EvoMLPNN}})$	*ME*	MCCT (sec.)
1	**SHCB**	−1.0316	1E-4	1000	100	−1.0316	−1.0316	−1.0316	0	12.32
2	**GP**	3	1E-4	1000	100	3	3	3	0	12.36
3	**SH**	−186.7309	1E-4	1000	100	−186.7309	−186.7309	−186.7309	0	12.32
4	**ZA_{10}**	0	1E-4	3000	100	9.861E-31	8.306E-29	4.175E-28	8.306E-29	54.07
5	**ZA_{30}**	0	1E-4	5000	100	1.889E-12	6.231E-07	9.341E-06	6.227E-07	155.32
6	**ZA_{50}**	0	1E-4	10000	100	6.437E-10	9.986E-06	7.569E-05	9.986E-06	451.12
7	**$Sphere_{30}$**	0	1E-4	5000	100	0	3.002E-28	2.945E-27	2.989E-28	151.77
8	**$Sphere_{50}$**	0	1E-4	10000	100	0	7.138E-28	9.584E-27	7.138E-28	450.52
9	**$Sphere_{100}$**	0	1E-4	10000	100	2.927E-28	8.361E-27	3.127E-26	8.361E-27	810.66

TABLE II. Comparison results (*ME*s) of the proposed EvoMLPNN method with those published algorithms

TP No.	Function name	NM-PSO [25]	PSACO [23]	GA-PSO [25]	DE-PSO [26]	AM-PSO1 [26]	AM-PSO2 [26]	CHA [23]	CTSS [23]	AIA-PSO [22]	RGA-PSO [22]	EvoMLPNN
2	**GP**	0.00003	0.00000000	0.00012	3.67E-5	3.81E-5	9.89E-6	0.0010	0.001	**0.00000000**	**0.00000000**	**0.00000000**
3	**SH**	0.00002	1.09239E-09	0.00007	3.03E-6	1.72E-5	2.54E-7	0.0050	0.001	**0.00000000**	**0.00000000**	**0.00000000**
4	**ZA_{10}**	–	2E-08	0.00000	6.91E-5	7.41E-5	9.11E-5	1E-6	–	**3.287E-42**	4.376E-26	8.306E-29
6	**ZA_{30}**	–	–	–	–	–	–	–	–	8.099E-06	4.537E-06	**6.227E-07**

APPENDIX

Six-hump Camel Back (SHCB) (two variables) [24]

$$f(\mathbf{x}) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2$$

search domain: $-3 \le x_1 \le 3;\ \ -2 \le x_2 \le 2$

one global minimum at two different points: $\mathbf{x}^* = (-0.0898, 0.7126)$ and $\mathbf{x}^* = (0.0898, -0.7126)$, $f(\mathbf{x}^*) = -1.0316$

Goldstein-Price (GP) (two variables) [23, 25]

$$f(\mathbf{x}) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right]$$

$$\times\left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$$

search domain: $-2 \le x_n \le 2,\ n = 1, 2$

four local minima; one global minimum: $\mathbf{x}^* = (0, -1)$, $f(\mathbf{x}^*) = 3$

Shubert (SH) (two variables) [23, 25]

$$f(\mathbf{x}) = \sum_{n=1}^{N-1}\left[100(x_n^2 - x_{n+1})^2 + (x_n - 1)^2\right]$$

search domain: $-10 \le x_n \le 10,\ \ n = 1, 2$

760 local minima; 18 global minima; $f(\mathbf{x}^*) = -186.7309$

Zakharov (ZA_n) (*N* variables) [23, 25]

$$f(\mathbf{x}) = \sum_{n=1}^{N} x_n^2 + \left(\sum_{n=1}^{N} 0.5nx_n\right)^2 + \left(\sum_{n=1}^{N} 0.5nx_n\right)^4$$

Three functions were considered: $ZA_{10}, ZA_{30}, ZA_{50}$

search domain: $-5 \le x_n \le 10,\ \ n = 1, 2, ..., N$

several local minima (exact number unspecified in usual literature);

global minimum: $\mathbf{x}^* = (0,\ldots,0)$, $f(\mathbf{x}^*) = 0$

Sphere (*N* variables) [14]

There functions were considered: $Sphere_{30}$, $Sphere_{50}$, $Sphere_{100}$

$$f(\mathbf{x}) = \sum_{n=1}^{N} x_n^2$$

search domain: $-5.12 \le x_n \le 5.12,\ n = 1, 2, ..., N$

one global minimum: $\mathbf{x}^* = (0, \ldots, 0)$, $f(\mathbf{x}^*) = 0$

REFERENCES

[1] M. De Lozzo, et al., "Multilayer perceptron for the learning of spatio-temporal dynamics—application in thermal engineering," Engineering Applications of Artificial Intelligence, vol. 26, no. 10, 2013, pp. 2270-2286.

[2] E. Kolay and T. Baser, "Estimating of the dry unit weight of compacted soils using general linear model and multi-layer perceptron neural networks," Applied Soft Computing, vol. 18, 2014, pp. 223-231.

[3] S. Mabu, et al., "Ensemble learning of rule-based evolutionary algorithm using multi-layer perceptron for supporting decisions in stock trading problems," Applied Soft Computing, vol. 36, 2015, pp. 357-367.

[4] A.F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," Neural Networks, vol. 6, no. 4, 1993, pp. 525-533.

[5] G.P. Rangaiah, Stochastic Global Optimization Techniques and Applications in Chemical Engineering: Techniques and Applications in Chemical Engineering, World Scientific Publishing Company, 2010.

[6] T.B. Ludermir and W.R. de Oliveira, "Particle swarm optimization of MLP for the identification of factors related to common mental disorders," Expert Systems with Applications, vol. 40, no. 11, 2013, pp. 4648-4652.

[7] H. Liu, et al., "An experimental investigation of two Wavelet-MLP hybrid frameworks for wind speed prediction using GA and PSO optimization," International Journal of Electrical Power & Energy Systems, vol. 52, 2013, pp. 161-173.

[8] J. Sun, et al., "Quantum-behaved particle swarm optimization with Gaussian distributed local attractor point," Applied Mathematics and Computation, vol. 218, no. 7, 2011, pp. 3763-3775.

[9] B. Li, et al., "Slope stability analysis based on quantum-behaved particle swarm optimization and least squares support vector machine," Applied Mathematical Modelling, vol. 39, no. 17, 2015, pp. 5253-5264.

[10] K. Hassani and W.S. Lee, "Multi-objective design of state feedback controllers using reinforced quantum-behaved particle swarm optimization," Applied Soft Computing, vol. 41, 2014, pp. 66-76.

[11] C.I. Sun, et al., "An improved vector particle swarm optimization for constrained optimization problems," Information Sciences, vol. 181, no. 6, 2011, pp. 1153-1163.

[12] Y. Li, et al., "Dynamic-context cooperative quantum-behaved particle swarm optimization based on multilevel thresholding applied to medical image segmentation," Information Sciences, vol. 294, 2015, pp. 408-422.

[13] O.E. Turgut, "Hybrid chaotic quantum behaved particle swarm optimization algorithm for thermal design of plate fin heat exchangers," Applied Mathematical Modelling, vol. 40, no. 1, 2016, pp. 50-69.

[14] R.P. Parouha and K.N. Das, "A memory based differential evolution algorithm for unconstrained optimization," Applied Soft Computing, vol. 38, 2016, pp. 501-517.

[15] R.M. Rizk-Allah, et al., "Hybridizing ant colony optimization with firefly algorithm for unconstrained optimization problems," Applied Mathematics and Computation, vol. 224, 2013, pp. 473-483.

[16] K. Hornik, et al., "Multilayer feedforward networks are universal approximators," Neural Networks, vol. 2, no. 5, 1989, pp. 359-366.

[17] H.S. Kim, et al., "Nonlinear dynamics, delay times, and embedding windows," Physica D: Nonlinear Phenomena, vol. 127, no. 1-2, 1999, pp. 48-60.

[18] X. Fan, et al., "Chaotic characteristic identification for carbon price and an multi-layer perceptron network prediction model," Expert Systems with Applications, vol. 42, no. 8, 2015, pp. 3945-3952.

[19] J. Sun, et al., Particle Swarm Optimization: Classical and Quantum Perspectives, CRC Press, 2012.

[20] J. Sun, et al., "Quantum-Behaved Particle Swarm Optimization: Analysis of Individual Particle Behavior and Parameter Selection," Evolutionary Computation, vol. 20, no. 3, 2012, pp. 349-393

[21] J.Y. Wu, "An improved quantum-behaved particle swarm optimization based multilayer perceptron classifier for medical data classification," Proc. The 15th Asia Pacific Industrial Engineering and Management Systems Conference, 2014, pp. 2219-2224.

[22] J.Y. Wu, "Solving unconstrained global optimization problems via hybrid swarm intelligence approaches," Mathematical Problems in Engineering, vol. 2013, 2013, pp. 1-15.

[23] P.S. Shelokar, et al., "Particle swarm and ant colony algorithms hybridized for improved continuous optimization," Applied Mathematics and Computation, vol. 188, no. 1, 2007, pp. 129-142.

[24] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs Springer, 1999.

[25] Y.T. Kao and E. Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions," Applied Soft Computing, vol. 8, no. 2, 2008, pp. 849-857.

[26] R. Thangaraj, et al., "Particle swarm optimization: Hybridization perspectives and experimental illustrations," Applied Mathematics and Computation, vol. 217, no. 12, 2011, pp. 5208-5226.