International Journal of Computational Intelligence Systems, Vol. 7, No. 5 (October 2014), 979-992

Neural Networks for Normative Knowledge Source of Cultural Algorithm

Vahid Seydi Ghomsheh
Artificial Intelligence Department, Islamic Azad University, Science and Research Branch, Tehran, Iran.
E-mail: v_seydi@azad.ac.ir
Mohamad Teshnehlab
Electrical Engineering, Control Department, K. N. Toosi University of Tech., Tehran, 19697, Iran.
E-mail : teshnehlab@eetd.kntu.ac.ir
Mahdi Aliyari Shoorehdeli
Mechatronics Department, K. N. Toosi University of Tech., Tehran, 19697, Iran.
E-mail:aliyari@eetd.kntu.ac.ir
Mojtaba Ahmadieh Khanesar
Electrical and Control Engineering Department, Semnan University, Semnan, Iran.
E-mail:ahmadieh@profs.semnan.ac.ir

Received 31 May 2013

Accepted 5 November 2013

Abstract

This study presents the normative knowledge source for the belief space of cultural algorithm(CA) based on an adaptive Radial Basis Function Neural Network (RBFNN). The use of the RBFNN makes it possible to use the previous upper and lower bounds of the normative knowledge to update them and to extract a logical relationship between the previous parameters of the normative knowledge and their new values. The proposed algorithm(N^3KCA) is similar to what the human brain does, i.e. to predict the new values of the bounds of normative knowledge based on the previous ones and some knowledge, which it has gained from the previous successive updates. Finally, the proposed cultural algorithm is evaluated on 10 unimodal and multimodal benchmark functions. The algorithm is compared with several other optimization algorithms including previous version of cultural algorithm. In order to have a fair comparison, the number of cost function evaluation is kept the same for all optimization algorithms. The obtained results show that the proposed modification enhances the performance of the CA in terms of convergence speed and global optimality.

Keywords: Cultural algorithm (CA), global optimization, knowledge Sources, normative knowledge, adaptive Radial Basis Function Neural Network (RBFNN).

1. Introduction

Optimization is an important issue in different scientific applications. Many researchers dedicated their studies to algorithms that can be used to find an optimal solution for different applications. Evolutionary computation techniques such as genetic algorithm, evolutionary strategy, and evolutionary programming and swarm intelligence algorithms such as particle swarm intelligence algorithm and ant colony optimization are powerful tools for solving optimization problems [1]-[4]. Similar to particle swarm optimization and ant algorithm in which members try to share their experiences, CA tries to model social intelligence based on natural cultural evolution to solve the optimization problems [5]-[6][24].

Facing with an engineering optimization problem with extensive domain knowledge that cannot be easily integrated into the population level, CA is a proposed optimization method. The CA Performance is studied in many benchmark optimization problems [7] and in a number of diverse application areas. For example the efficacy of a version of CA is tested on Loney's solenoid design which is an electromagnetic engineering problem [8]. In [9] authors applied the CA to the pressure vessel optimization problem which is a benchmark engineering design problem. The CA is also applied to some other engineering applications such as prediction by functional link-based neural fuzzy network [10], optimization of the tension compression of a spring weight [11] and modeling the evolution of agriculture [12].

Three major components of CA are population space, belief space and a protocol that defines the relationship between the population and the belief space. The population space can be based on any population-based computing models such as genetic algorithm, evolutionary programming and particle swarm optimization method [10]. The belief space stores and updates the knowledge acquired from the experience of individuals in the population space. By using this knowledge, the belief space conducts the population to the optimal solution. This mutual interaction between the population and the belief space will continue until the stop criteria of the algorithm are visited. One of these knowledge sources is normative knowledge which determines the upper and lower bounds for each variable. The normative knowledge source identifies promising variable ranges of the solutions.

This study presents a novel idea presenting the normative knowledge source, in the belief space of cultural algorithm; which is named "Neural Networks for Normative Knowledge Source of Cultural Algorithm"(N^3KCA). The proposed novel normative knowledge source benefits from an adaptive RBFNN. RBFNN has been widely used in many areas, such as data mining, pattern recognition, signal processing, time series prediction and nonlinear system modeling and control [13-23]. It is suggested that there exist a relationship between the previous upper and lower bounds of the normative knowledge and their new values. Using RBFNN as a normative knowledge source, it is possible to extract this logical relationship. The proposed method tries to pretend the way human brain thinks about the upper and the lower bounds of variables, considering their histories. Finally, the proposed cultural algorithm is evaluated on 10 unimodal and multimodal benchmark functions. The algorithm is compared with several other optimization algorithms such as previous version of cultural algorithm, differential evolution, particle swarm optimization and genetic algorithm. In order to have a fair comparison, the number of cost function evaluation is kept the same for all optimization algorithms. The obtained results show that CA with normative knowledge source based on RBFNN outperforms these algorithms in terms of convergence speed and global optimality.

The rest of this study is organized as follows. The components of CA are reviewed in section II. Section III presents normative knowledge source based on an adaptive RBFNN approach in details. Section IV compares the N^3KCA algorithm with existing CA and various optimization algorithms for a set of benchmark functions. Discussions and further investigations on the N^3KCA are made in this section. Final conclusion is presented in section V.

2. Cultural algorithm overview

In this section we describe the traditional CA. The key idea of CA is to store and update the problem solving knowledge with the feedback from the population and to guide the search using this knowledge [24]. The components of CA are population, belief space, acceptance function, and influence function. These major components of CA are depicted in Fig. 1.

2.1. *The belief* space

The experience of individuals are used and stored in information repository called belief space. These experiences can be used by other individuals. In other words the members of the population share their experiences in the belief space and subsequently the knowledge is extracted from these experiences. The benefit of CA over other evolutionary algorithms is that other than sharing the information with offspring the information is shared with other members of the group. CA employs sets of knowledge sources which are characterized by their appearing in the problem solving process. Reynold and Ali [9] identified five basic categories of Knowledge. Each of which are added in different time to achieve a various problem solving capabilities [7][25][26]. These five knowledge sources are normative knowledge, situational knowledge, domain knowledge, history knowledge and topographical knowledge. The range of acceptable behaviors in each generation is represented by normative knowledge [7]. Situational knowledge keeps exemplars of successful solutions. Relationships and interactions between the objects in the domain are kept in the domain knowledge source [25]. Temporal and special patterns of behavior are stored in history and topographical knowledge sources respectively [25][26]. Any cultural knowledge can be expressed as some combination of these five knowledge sources [9]. The goal of this study is to present a novel normative knowledge source and to meet this goal normative and situational knowledge sources are employed.

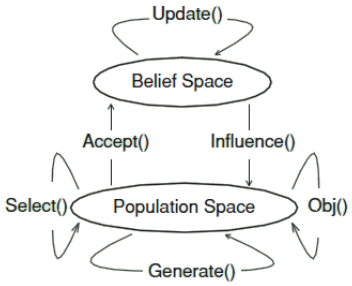

Fig1. The CA framework[9]

The formal syntax of belief space defined in this paper is $<N[n],S>$ where N is a normative knowledge source and includes the interval information for each n variable and S is a situational knowledge component.

2.1.1. The existing normative knowledge source

In all pervious works $N[j]$ is represented as $N[j] =< I_j, L_j, U_j >$. I_j is the closed interval of j-th variable, $I_j = [l_j, u_j] = \{x \mid l_j \le x \le u_j, x \in \mathbb{R}\}$ where for the j-th variable the lower bound, l_j, and upper bound, u_j, are initialized as the domain values and they can be changed later. L_j and U_j denote the performance score of the lower and upper bounds of j-th variable, respectively. Also in previous studies the normative knowledge update mechanism is as follows. Assume that the i-th individual affects the lower and upper bounds of the j-th variable at generation t. The update formulas are given below.

$$l_j^{t+1} = \begin{cases} x_{i,j}^t & if \;\; x_{i,j}^t < l_j^t \;\; or \;\; f(x_i^t) < L_j^t \\ l_j^t & Otherwise \end{cases} \quad (1)$$

$$L_j^{t+1} = \begin{cases} f(x_i) & if \;\; x_{i,j} < l_j^t \;\; or \;\; f(x_i) < L_j^t \\ L_j^t & Otherwise \end{cases} \quad (2)$$

$$u_j^{t+1} = \begin{cases} x_{i,j}^t & if \;\; x_{i,j}^t > u_j^t \;\; or \;\; f(x_i^t) < U_j^t \\ u_j^t & Otherwise \end{cases} \quad (3)$$

$$U_j^{t+1} = \begin{cases} f(x_i) & if \;\; x_{i,j} > u_j^t \;\; or \;\; f(x_i) < U_j^t \\ U_j^t & Otherwise \end{cases} \quad (4)$$

Where $x_{i,j}^t$ represents the j-th variable of i-th individual at generation t and $f(x_i)$ denotes the fitness function for it.

2.1.2. The situational knowledge source

Some good experiences of individuals are kept in this knowledge source. Situational knowledge guides individuals to move towards these elite experiences. This is the earliest knowledge source used with the CA which is inspired by elitist approaches in genetic algorithm. The basic idea of situational knowledge source is also very similar to the movement of particles towards the global best in particle swarm optimization (PSO). This knowledge is updated as follows.

$$S^{t+1} = \begin{cases} x_{best}^t & if \;\; f(x_{best}^t) < f(S^t); \\ S^t & Otherwise \end{cases} \quad (5)$$

where x_{best}^t is the best individual in the population at current generation t.

2.2. *The population space*

The population space consists of a set of possible solutions for the problem, and can be modeled using any population-based approach. The population model used here is a simple evolutionary algorithm where each individual is a vector of real-valued variables. In each generation an individual is evolved by the mutation operators using a specific knowledge source. Each knowledge source specifies a different mutation operator.

In this study population space is called P_{Main}. Each individual in P_{Main} has a number of features that are problem variables. Therefore when we want to optimize a function with n variables, i-th individual in P_{Main} can be considered as follows.

$$\vec{x}_i = [x_1 \;\; x_2 \;\; \dots \;\; x_n] \quad (6)$$

2.3. *The acceptance function*

The acceptance function determines which of individuals and their behaviors can impact the belief space knowledge. The number of individuals which impact the belief space can range between 1% and 100% of the population size, based on selected criteria.

2.4. *The influence function*

The belief space can influence the new individuals of the population space according to the knowledge sources influence function. This function defines the method by which the knowledge in the belief space controls the mutation operator in the population space. As mentioned earlier, different mutation operators are defined by each knowledge source. Since in current research only two sources of situational knowledge and normative knowledge are used, therefore influence functions are defined as two

sources of knowledge. The mutation operator according to situational knowledge is defined as follows.

$$x_{i,j}^{t+1} = \begin{cases} x_{i,j}^{t} + \left|(x_{i,j}^{t} - S_j).N(0,1)\right| & if\ x_{i,j}^{t} < S_j \\ x_{i,j}^{t} - \left|(x_{i,j}^{t} - S_j).N(0,1)\right| & if\ x_{i,j}^{t} > S_j \\ x_{i,j}^{t} + b.\left|(x_{i,j}^{t} - S_j)\right|.N(0,1) & otherwise \end{cases} \quad (7)$$
$$j = 1,...,n$$

where n is the number of parameters; $x_{i,j}$ is the j-th element in the i-th individual at generation t; $\beta(0.01 \sim 0.6)$ is a constant that is selected equal to 0.3 as used by Chung [27], S_j is the j-th parameter in the best individual (S), and $N_{i,j}(0,1)$ is a random number with normal probability distribution function. The normative knowledge defines the following mutation operator.

$$x_{i,j}^{t+1} = \begin{cases} x_{i,j}^{t} + \left|(u_j - l_j).N(0,1)\right| & if\ x_{i,j}^{t} < l_j \\ x_{i,j}^{t} - \left|(u_j - l_j).N(0,1)\right| & if\ x_{i,j}^{t} > u_j \\ x_{i,j}^{t} + \beta.\left|(u_j - l_j)\right|.N(0,1) & otherwise \end{cases} \quad (8)$$
$$j = 1,...,n$$

3. RBFNN structure as normative knowledge source

In this section we introduce the normative knowledge-generating system. As mentioned earlier, the normative knowledge produces the lower and upper bounds for each dimension of an optimization problem which is led to the optimal variables range. Considering an optimization problem with n variables, a normal RBFNN is designed to generate the n –data pairs which represent the previous lower and upper bounds for variables. The structure of a normal RBFNN with m neurons is shown in Fig. 2.

3.1. *The inputs of normal RBFNN*

As can be seen from Fig. 2, this neural network has four inputs which are the index number of the feature, the feature value, the lower and upper bounds of the feature. For determining the input x^t of this artificial neural network, accepted individuals in P_{Main}, participate in a discrete recombination which benefits from a roulette wheel mechanism. This individual which is produced from the accepted individuals is called Recombined Accepted Individual (*RAI*). As Fig. 2 shows the features of the *RAI* are fed to the network separately along with x^t, l^t and u^t. In this way in order to compute the normative knowledge source, this neural network is evaluated n times. In general, the network equation can be expressed as follow.

$$\vec{Y} = \sum_{i=1}^{m} \vec{W}\,\vec{\phi}_i(\vec{X}, \vec{\sigma}_i, \vec{c}_i) \quad (9)$$

where m is the number of neurons. The neural network input matrix at generation number t is as follows.

$$\vec{X} = \begin{bmatrix} i_1 & RAI_1^t & l_1^t & u_1^t \\ i_2 & RAI_2^t & l_2^t & u_2^t \\ \vdots & \vdots & \vdots & \vdots \\ i_n & RAI_n^t & l_n^t & u_n^t \end{bmatrix} \quad (10)$$

which i_k *(k=1,2,…,n)* is the index of variables that is normalized to the variable range, RAI_k^t is value of k-th variable at generation t, and l_k^t and u_k^t are the lower and upper bounds for k-th variable in generation t respectively. In each stage, the neural network is applied to a row of this matrix. The network output is as follows.

$$\vec{Y} = \begin{bmatrix} l_1^{t+1} & u_1^{t+1} \\ l_2^{t+1} & u_2^{t+1} \\ \vdots & \vdots \\ l_n^{t+1} & u_n^{t+1} \end{bmatrix} \quad (11)$$

3.2. *The feedforward algorithm* of *normal RBFNN*

Obviously, in every stage of the feedforward calculation of the neural network, a row of $\vec{Y}$ is achieved as the output of the neural network. So in order to find the lower and upper bounds for all n variables, it is required to run the neural network feedforward algorithm for n times.

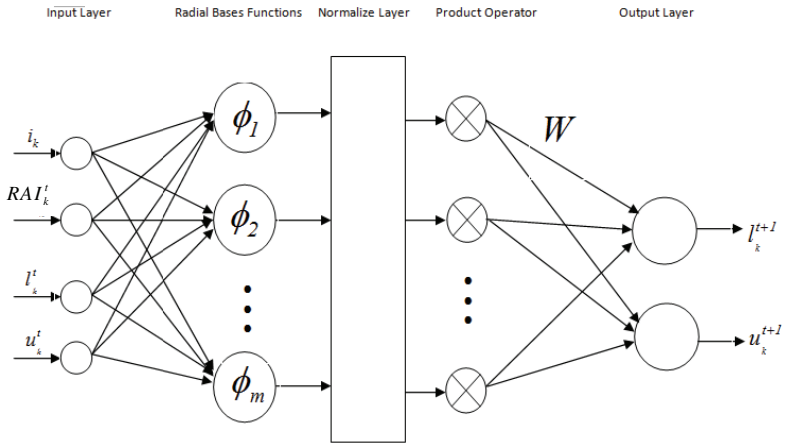

Figure 2. The structure of Normal Radial Bases Function Neural Network with m neuron for normative knowledge source.

The output layer weights for m neuron are as follows.

$$\vec{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \end{bmatrix} \tag{12}$$

These weights should be trained to achieve their optimal values during the process. The normalized output of each radial basis function is calculated as follows.

$$\phi_i(\vec{X},\vec{\sigma}_i,\vec{c}_i) = \frac{\varphi_i(\vec{X},\vec{\sigma}_i,\vec{c}_i)}{\sum_{j=1}^{m}\varphi_j(\vec{X},\vec{\sigma}_j,\vec{c}_j)},\ i=1,2,...,m \tag{13}$$

where $\varphi_i(\vec{X},\vec{\sigma}_i,\vec{c}_i)$ is the Gaussian radial basis functions as:

$$\varphi_i(\vec{X},\vec{\sigma}_i,\vec{c}_i) = \exp\left[-0.5\left(\left(\frac{i_k^t - c_{i1}}{\sigma_{i1}}\right)^2 + \left(\frac{x_k^t - c_{i2}}{\sigma_{i2}}\right)^2 + \left(\frac{l^t - c_{i3}}{\sigma_{i3}}\right)^2 + \left(\frac{u^t - c_{i4}}{\sigma_{i4}}\right)^2\right)\right] \tag{14}$$

As the relations above show, there are two groups of parameters in this structure that should be optimized in the process of the training artificial neural network. These two groups of the parameters related to the radial functions, namely:

$$\vec{c} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ \vdots & \vdots & \vdots & \vdots \\ c_{m1} & c_{m2} & c_{m3} & c_{m4} \end{bmatrix} \tag{15}$$

$$\vec{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} & \sigma_{24} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{m1} & \sigma_{m2} & \sigma_{m3} & \sigma_{m4} \end{bmatrix} \tag{16}$$

where $\vec{c},\vec{\sigma}$ are the matrix of center and standard deviations of the radial bases functions(RBFs) respectively. The output layer weight vector of the neural network is specified as$\vec{W}$. Since there is not any target values for the lower and upper bounds, gradient based estimation methods cannot be used. In fact, there is no neural network error to optimize the neural network parameters. The derivative free based methods, such as evolutionary algorithms [31]-[34] and swarm intelligence algorithms [35]-[41] are independent from objective function with respect to neural network parameters and hence can be used, to optimize the parameters of neural network. We use the PSO to optimize the output weights $\vec{W}$ and EP to optimize the center values of the radial basis functions. In order to optimize $\vec{W}$ parameters, and the center parameters, two populations are considered as P_W and P_C. The values of variance are selected as one third of distance between the centers in each dimension and these parameters are not trained during the optimization.

3.3. *Training $\vec{W}$ parameters*

We use PSO algorithm for the training of $\vec{W}$. In this algorithm each particle in P_W is a vector of $m \times 2$ variables. These weights are initialized with random

numbers. In order to calculate the best personal and the global experiences, we need to evaluate these particles. In order to evaluate a particle, first we calculate the lower and upper bounds corresponding to this particle by neural network feedforward algorithm. Using the achieved lower and upper bounds, RAI is mutated by normative knowledge source influence function. This procedure is done for all particles in P_W and subsequently the fitness of the corresponding mutated RAI is considered as the fitness of each particle.

It should be noted that the centers of *RBFs* which are used to train $\vec{W}$, are selected as the best member of P_C . The optimization of P_C is described as follows.

3.4. *Training categories centers parameters C*

In order to obtain the best values for the parameters C , evolutionary programming is utilized, which benefits from a simple mutation. The centers of the *RBFs* are uniformly distributed in each dimension. The members of P_C are selected as different permutations of these uniformly distributed centers. Thus each member is defined by a matrix, whose rows are centers of a neuron and its columns include different dimensions of the inputs space which is considered to be 4 (see equation 15). We define a mutation operator on the members of P_C that changes the places of the centers within the same dimension between different neurons. In other words, no new center values of the *RBFs* are generated and the values of the offspring are a permutation of its parent. This simple mutation will considerably lessen the computational cost. Using the mutation operator the offspring are generated. In order to evaluate a member, first we calculate the lower and upper bounds corresponding to this member using neural network feedforward algorithm. It should be noted that the output weights of neural network are the same for all members and are equal to the global best particle of P_W . Using the achieved lower and upper bounds, RAI is mutated by normative knowledge source influence function. This procedure is repeated for all members in P_C and its offsprings. The fitness of the corresponding mutated RAI is considered as the fitness of each member. A selection operator chooses $|P_C|$ (the number of members of P_C) from the best members of the next generation. This procedure is repeated in each generation; In this way, with simultaneous optimization of RBFs centers and output weights, we obtain the best configuration for RBFs and neural network parameters.

3.5. *The Pseudo code of the whole algorithm*

The Pseudo code of the described algorithm can be briefly presented as follows.

1. Initialize randomly P_{Main} ، P_W and P_C .
2. Iterate the following steps until the stop criteria are visited.

2.1. sort the P_{Main} based on fitness function.
2.2. update situational knowledge as equation 5
2.3. Select individuals from P_{Main} considering acceptance function and generate RAI as described in section 3.1.
2.4. Perform the procedure described in section 3.3 for one generation and update the particles in P_W .
2.5. Perform the procedure described in section 3.4. for one generation and update the particles in P_C .
2.6. update normative knowledge using *RBFNN* (the best individual in P_C would be replaced by C and the best particle in P_W would be replaced by W).
2.7. Mutate Individuals in P_{Main} using the normative knowledge and situational knowledge Influences function (50% by each).

4. Simulation result

In this section, some experiments are carried out on the benchmark functions to evaluate the proposed CA which benefits from a RBFNN normative knowledge source. In addition, the proposed method is compared with existing structures for The CA and The other optimization methods e.g. particle swarm optimization (PSO)[42]-[44], The genetic algorithms(GA) [45]-[47]and The differential evolution(DE)[42],[48],[49].

Table 1. Ten Benchmark functions are used in this paper. The first five functions are unimodal and the remaining are multimodal

Name of function	Test function	Search space	Global f_{min}
Sphare	$f_1(x)=\sum_{i=1}^{D} x_i^2$	$[-100,100]^D$	$\vec{x}=0, f(\vec{x})=0$
Schwefel2.21	$f_2(x)=\max_i(\lvert x_i\rvert)$	$[-100,100]^D$	$\vec{x}=0, f(\vec{x})=0$
Rosenbrock	$f_3(x)=\sum_{i=1}^{D-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	$[-30,30]^D$	$\vec{x}=1, f(\vec{x})=0$
Quadric Noise	$f_4=\sum_{i=1}^{D} i x_i^4+\text{rand}[0,1)$	$[-1.28,1.28]^D$	$\vec{x}=0, f(\vec{x})=0$
Step	$f_5(x)=\sum_{i=1}^{D}\lfloor x_i+0.5\rfloor^2$	$[-100,100]^D$	$\vec{x}=0, f(\vec{x})=0$
Griewank	$f_6(x)=\frac{1}{4000}\sum_{i=1}^{D} x_i^2-\prod_{i=1}^{D}\cos(\frac{\pi_i}{\sqrt{i}})+1$	$[-600,600]^D$	$\vec{x}=0, f(\vec{x})=0$
Rasterigin	$f_7=\sum_{i=1}^{D}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$[-5.12,5.12]^D$	$\vec{x}=0, f(\vec{x})=0$
Ackley	$f_8(x)=-20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}}-e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}+20+e$	$[-32,32]^D$	$\vec{x}=0, f(\vec{x})=0$
Generalized Penalized	$f_9(x)=\frac{\pi}{D}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_D-1)^2\right\}+\sum_{i=1}^{D} u(x_i,10,100,4)$, where $y_i=1+\frac{1}{4}(x_i+1)$, $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	$[-50,50]^D$	$\vec{x}=0, f(\vec{x})=0$
Non continuous Rasterigin	$f_{10}=\sum_{i=1}^{D}\left[y_i^2-10\cos(2\pi y_i)+10\right]$, where $y_i=\begin{cases}x_i & \lvert x_i\rvert<0.5\\ \frac{\text{round}(2x_i)}{2} & \lvert x_i\rvert>0.5\end{cases}$	$[-5.12,5.12]^D$	$\vec{x}=0, f(\vec{x})=0$

4.1. *Benchmark functions and algorithm configuration*

We have chosen ten test functions that are widely used in the nonlinear global optimization literature [28-30]. Function names, formulas, range of variables and the global optima are listed in *Table 1* these benchmark functions have a wide variety of the different landscapes and present significant challenges to the optimization methods. The step function is a discontinuous unimodal function, but Sphere, Schwefel's 2.21, Rosenbrock, and Quadric Noise are continuous unimodal functions. Griewank, Rastring, Ackley and Generalized Penalized are difficult multimodal functions where the number of local optima increases exponentially with the problem dimension. "Non-continuous Rasterigin" is also a discrete multi-modal function.

In contemplation of comparing the proposed algorithm, the standard version of existing CA, PSO, GA and DE algorithms which are widely applied to optimization applications, are used. The existing CA uses two knowledge sources: situational and normative knowledge. This algorithm uses

evolutionary programming model for its population space. In addition, it benefits from influence functions operators defined by the knowledge sources mentioned above, each one having a 50% chance to be applied to an individual. DE is a simple method that uses the difference between two solutions to probabilistically adapt a third solution. For DE, we use a weighting factor $F \sim U(0.2, 0.8)$ and a crossover probability P_c=*0.2*. GA is based on natural selection in the theory of biological evolution. Here, we use real value coding for variables, roulette wheel selection, arithmetic crossover with a crossover probability P_c=*0.7*, uniform mutation with a mutation probability P_m= *0.1* and elitism operator that uses the best chromosome at each generation for the next generation. The PSO algorithm is based on the swarming behavior of birds and fish. For PSO we use only global learning (star topology), the inertia weight *w* decreases linearly from 0.9 to 0.4 and the acceleration coefficients are set as $c_1 = c_2 = 2$ same as the common configuration in a standard PSO.

The value of maximum velocity is set as the scope of the search space for each case. The values of the parameters for above mentioned algorithms are shown in details in *Table 2*.

Two trends are used for determining the population size. In the first set of experiences, the sizes of population in all algorithms are kept constant and equal to 10. In the second set of experiments in order to have a fair comparison, the population sizes of the algorithms are considered in such a way that the orders of complexity of all algorithms are kept the same.

The evaluation of the fitness function is considered as the key operator, for determining the algorithm complexity. In other words the other operations of algorithms are considered to be neglectable when compared to the evaluation of the fitness function. As a result the complexity of these algorithms can be calculated as in *Table 3*.

In addition, if the sizes of all populations are considered equal as $|P_{Main}| = |P_W| = |P_C| = PopSize$, then table 3 is updated as *Table 4*.

As *Table 4* illustrates, in order to achieve the same complexity for N^{3}KCA v.s CA, PSO, GA and DE ($\theta_{N^3KCA} = \theta_{CA} = \theta_{PSO} = \theta_{GA} = \theta_{DE}$) the population size of N^{3}KCA is set to 10 while for CA, PSO and GA the size is set to 30 and for DE is set to 15.

4.2. *Comparisons on the solutions*

Table 5. shows the comparison results of N^{3}KCA and previous approaches. The dimension of the set of functions is set to 30 and the simulations are repeated for 15 times. As can be seen from the *Table 5* the proposed algorithm outperforms others, when they are applied to f_2, f_3 and f_4. It should be noted that since the complexity in terms of number of cost function evaluation in the proposed algorithm is higher than other methods, the population size of other algorithms are considered bigger to achieve the same complexity. But if the population sizes of the algorithms are kept constant then the proposed algorithm performs better in eight of ten functions. It is also noticeable, in the functions that the proposed algorithm is not the best; it is often the second best algorithm.

Table 2. detail of all algorithms used in the comparison

CA	knowledge sources=normative knowledge and situational knowledge influence function =see equation 7 and 8(50% by each one) acceptance function=decrease linearly from 70% to 1%(one individual) of the population
PSO	c1=2 c2=2 w= decreases linearly from 0.9 to 0.4
GA	selection : roulette wheel crossover : arithmetic crossover mutation : uniform mutation Pc=0.7 Pm=0.1
DE	weighting factor $F \sim U(0.2, 0.8)$ crossover probability=0.2

Table 3. The complexity of all algorithms used in the comparison

$T_{N^3KCA}(\lvert P_{Main}\rvert, \lvert P_W\rvert, \lvert P_C\rvert, MaxGen) = MaxGen \times (\lvert P_{Main}\rvert + \lvert P_W\rvert + \lvert P_C\rvert)$
$T_{CA}(\lvert P_{Main}\rvert, MaxGen) = MaxGen \times \lvert P_{Main}\rvert$
$T_{PSO}(\lvert P_{Main}\rvert, MaxGen) = MaxGen \times \lvert P_{Main}\rvert$
$T_{GA}(\lvert P_{Main}\rvert, MaxGen) = MaxGen \times \lvert P_{Main}\rvert$
$T_{DE}(\lvert P_{Main}\rvert, MaxGen) = MaxGen \times \lvert 2P_{Main}\rvert$

Table 4. The complexity of all algorithms used in the comparison. The sizes of all populations are kept as $\lvert P_{Main}\rvert = \lvert P_W\rvert = \lvert P_C\rvert = PopSize$

$T_{N^3KCA}(PopSize, MaxGen) = 3 \times MaxGen \times PopSize$
$T_{CA}(PopSize, MaxGen) = MaxGen \times PopSize$
$T_{PSO}(PopSize, MaxGen) = MaxGen \times PopSize$
$T_{GA}(PopSize, MaxGen) = MaxGen \times PopSize$
$T_{DE}(PopSize, MaxGen) = 2 \times MaxGen \times PopSize$

Although other algorithms are comparable with the proposed method in low dimensions, the performance improvement of the proposed algorithm in higher dimension is more significant.

The simulation results of the comparison between N^3KCA and the previous methods in higher dimensions are given in *Table 6*. This table shows the results of 15 independent runs of each algorithm in terms of the "best", mean and overall standard deviations (SD) of the solutions. Boldface in the table indicates the best result among other contenders. As can be seen from the table, when the dimension of the functions increases, the proposed algorithm outperforms previously mentioned algorithms. As can be seen from the table, the performance of the proposed method is significantly better than other methods in both unimodal and multimodal optimization functions. Especially the difference of the optimal solutions found in f_3, f_6, f_7 and f_{10} for the proposed algorithm with respect to other studied methods is much higher.

Furthermore, the N^3KCA can successfully jump out of the local minima on all of the multimodal functions and surpasses all the other algorithms on functions f_7, f_8, and f_9, when the dimension of the problem is high. Considering the fact that the global optimum of f_7 (Schwefel's function) is far away from any of the local optima, and the globally best solutions of f_8 and f_9 (continuous/noncontiguous Rastrigin's functions) are surrounded by a large number of local optima, it can be concluded that when the dimension of the problem is high, the proposed method has a great ability to avoid being trapped in the local optima and achieving global optimal solutions to multimodal functions. Hence, when facing the high dimensional optimization problems, the proposed algorithm is a viable choice.

Table 5. Search Result Comparison of N^3KCA, CA, PSO, GA, DE And Ten Test functions. The dimension of the functions is set to 30 and the simulations are repeated for 15 times.

Algorithm		N^3K CA	CA		PSO		GA		DE	
No of Population		10	10	30	10	30	10	30	10	15
f_1	Best	3.93E-05	8417.978	**6.41E-24**	7639.522	1100.535	24.37858	10.48305	0.000393	1.1E-14
	Average	0.003462	19801.03	**5.05E-15**	13558.66	6417.345	56.19411	23.35973	51.79014	3.01E-13
	STD	0.00795	9427.495	1.45E-14	3860.135	4130.721	21.01692	6.294131	79.60155	6.62E-13
f_2	Best	**0.131758**	47.07699	18.77534	30.08316	22.39965	3.352188	2.319512	12.66573	2.631799
	Average	**0.910074**	68.7028	40.47477	43.05844	35.50529	5.997855	2.775654	25.27999	8.188588
	STD	1.142646	11.55835	10.62628	9.566417	8.010516	1.559576	0.461447	8.095543	5.763398
f_3	Best	**20.86967**	14000058	4.00031	3242712	454565.9	575.7049	408.444	102.0368	21.70866
	Average	**53.94288**	38585206	195.1178	11875061	3759853	4876.847	5397.628	158494.7	1130.938
	STD	75.1631	32249031	326.1038	5612486	4024922	5018.659	11000	376828.9	4109.979
f_4	Best	**0.002555**	11.75635	0.011568	1.763573	0.485143	0.041842	0.019812	0.019513	0.016842
	Average	**0.013913**	29.34757	0.101171	6.064976	1.559537	0.087164	0.03512	0.179901	0.026718
	STD	0.014	15.89923	0.079775	5.527126	1.358883	0.028879	0.011958	0.356727	0.00545
f_5	Best	**0**	10034	**0**	7487	1442	18	12	0	**0**
	Average	2	17899.13	2.6	14462.47	7271.133	71.73333	23.53333	113.2	**0.066667**
	STD	2.299068	7850.139	6.663332	4626.26	4115.836	38.74545	6.57774	201.7889	0.2582
f_6	Best	0.000504	76.7618	0	69.75569	10.90482	1.219407	1.094347	0.008531	**3.31E-13**
	Average	0.076971	179.8822	0.013811	123.0279	58.7561	1.505747	1.210238	0.739918	**0.000329**
	STD	0.141774	84.22516	0.013342	34.74121	37.17648	0.189153	0.056648	0.855637	0.001273
f_7	Best	15.04736	181.2648	78.6016	208.3556	145.5899	67.57997	45.50053	**7.866826**	16.40774
	Average	44.19348	237.7617	140.8192	244.105	190.0289	86.77335	68.11653	**17.07578**	39.45923
	STD	20.48986	36.14612	35.93035	21.62167	21.34706	15.87286	15.78489	5.801352	14.03291
f_8	Best	0.011503	15.52469	2.958087	14.2152	11.13221	2.895991	2.07254	0.000562	**3.24E-08**
	Average	1.700451	18.22393	7.577175	16.66763	14.18926	3.398749	2.384077	1.809054	**3.28E-07**
	STD	1.010621	1.281967	3.891022	1.35747	1.880678	0.347746	0.225495	1.130469	5.18E-07
f_9	Best	1.77E-06	1294841	1.25E-13	84144.02	12.98313	1.745092	0.856514	0.251774	**1.24E-15**
	Average	0.346674	43318410	1.81378	6612810	3187647	3.769537	2.935819	710.5996	**0.133309**
	STD	0.712239	38858718	3.498671	8270429	9257429	1.704181	2.033324	1950.43	0.356261
f_{10}	Best	**21.00065**	131.6282	49	179.1644	139.3256	47.63865	48.36197	20.59906	27.10211
	Average	55.10014	219.7189	82.33333	224.4698	176.4306	74.29062	86.46813	28.36417	**33.8463**
	STD	26.6007	42.63726	18.4765	36.53753	48.03541	12.84533	23.85494	5.797934	4.764468

Fig. 3 and Fig. 4 depict the convergence graph of the studied algorithms. These figures show that the convergence speeds of the proposed algorithm when it is applied to different unimodal and multimodal functions, is much higher than the other algorithms. These figures also imply that if the generation number of the algorithms is set to a small value, the proposed method noticeably outperforms other methods. For example if the generation number for all algorithms is set to 100, almost none of the algorithms can achieve comparable results with respect to the N^{3}KCA.

Fig. 5 illustrates the Frobinous norm of the neural network weights used in the normative knowledge source of the proposed algorithm. As can be seen from the figure the weights of this neural network are trained to obtain their optimal values during the optimization process. Fig. 5 shows that in the multimodal optimization problem, the weights need to be changed more, in order to obtain an optimal result. In addition, it can be seen that the weights of neural network are generally more rapidly converge when the cost function is unimodal. Using this neural network, make it possible to obtain an optimal nonlinear and intelligent relationship between the previous values of the normative knowledge and its successful future values. The fact that the proposed algorithm obtains the best results especially in high dimensions shows that the proposed method is successful in using its experiences to obtain the desired knowledge source.

Table 6. Search Result Comparison of N^{3}KCA, CA, PSO, GA, DE And Ten Test functions. The dimension of the functions is set to 100 and the simulations are repeated for 15 times.

Algorithm		N^{3}K CA	CA		PSO		GA		DE	
No of Population		١٠	10	30	10	30	10	30	10	15
f_1	Best	**0.006677**	131970.6	194.6648	35708.52	27956.64	5237.26	1053.269	147.4437	0.234724
	Average	**4.730868**	172048	3121.186	57048.88	42477.02	7186.42	1874.682	1375.528	16.72613
	STD	3.134215	23748.68	3864.554	19467.92	11704.7	1574.104	396.8235	1044.489	35.29168
f_2	Best	**0.680019**	87.36373	80.29721	42.27448	34.70838	50.70327	20.6777	53.51162	55.88381
	Average	**2.170413**	93.20505	85.515	56.72238	47.98826	61.37755	28.91107	63.11614	64.64682
	STD	0.933936	2.554767	2.616001	8.197084	6.510296	7.656064	5.483909	6.421677	5.863988
f_3	Best	**93.30936**	5.12E+08	57456.61	28820015	9986870	1176026	133556.4	451853.9	1757.679
	Average	**293.0077**	7.43E+08	5200300	76430658	45115764	2920772	326489.1	4257928	74011.52
	STD	446.892	1.27E+08	10295630	43703547	30643107	1705872	114017.5	4147288	174356
f_4	Best	**0.005013**	849.7301	4.93921	50.89253	14.296	4.017097	0.352146	2.517897	0.349747
	Average	**0.026627**	1196.624	21.52691	116.9386	41.83727	7.506062	1.556055	7.710071	0.96677
	STD	0.027368	182.4113	19.13405	63.6448	29.87847	3.228422	0.682821	5.879923	0.734905
f_5	Best	**0**	118541	6395	36194	26481	6040	1281	633	1
	Average	**28.4**	170924.5	15485.6	56634.87	43672.93	7995.667	1925.467	2456.2	86.66667
	STD	18.74567	27166.16	6479.059	19157.24	14071.25	1196.1	436.15	1597.94	125.8478
f_6	Best	**0.23078**	1188.735	2.752549	322.3767	252.6098	48.13534	10.47942	2.326993	1.000176
	Average	**0.867852**	1549.369	29.06927	514.4399	383.2932	65.67778	17.87214	13.37975	1.150311
	STD	0.311573	213.7883	34.72286	175.2601	105.2226	14.16693	3.571411	9.400402	0.317737
f_7	Best	**7.584223**	1106.751	562.2261	833.4902	788.6953	792.8895	662.5668	328.0291	542.2767
	Average	**97.72475**	1313.261	665.5601	1008.288	883.7358	896.4352	788.5333	429.6297	585.3754
	STD	79.71627	103.8822	69.62043	91.90354	61.38746	78.17053	78.02651	70.21832	26.42879
f_8	Best	**0.012537**	20.04262	18.52987	15.30085	15.13416	9.326511	5.307857	3.79213	0.087751
	Average	**0.372472**	20.37859	19.15834	17.50186	16.4221	11.97795	6.130698	6.363929	0.557159
	STD	0.661231	0.17619	0.363509	1.128633	0.967738	3.442238	0.500685	1.389185	0.5149
f_9	Best	**1.05E-05**	1.00E+09	1.19E+01	2.68E+06	6.95E+05	1.67E+04	3.50E+01	1.01E+05	2.52E+00
	Average	**0.046205**	1.78E+09	47045.22	63548388	18987533	444169.2	350.4017	4521558	68877.38
	STD	0.102879	3.82E+08	101980.4	67823300	22158520	487852.4	605.7422	4711688	225166.6
f_{10}	Best	**16.84896**	9.79E+02	5.52E+02	7.80E+02	6.70E+02	7.30E+02	6.88E+02	3.53E+02	4.48E+02
	Average	**291.3551**	1212.644	713.2667	958.8124	831.5621	936.7113	815.8514	439.4116	530.6472
	STD	190.3396	112.0141	85.64104	98.00615	86.83269	90.17898	68.74915	48.439	43.24006

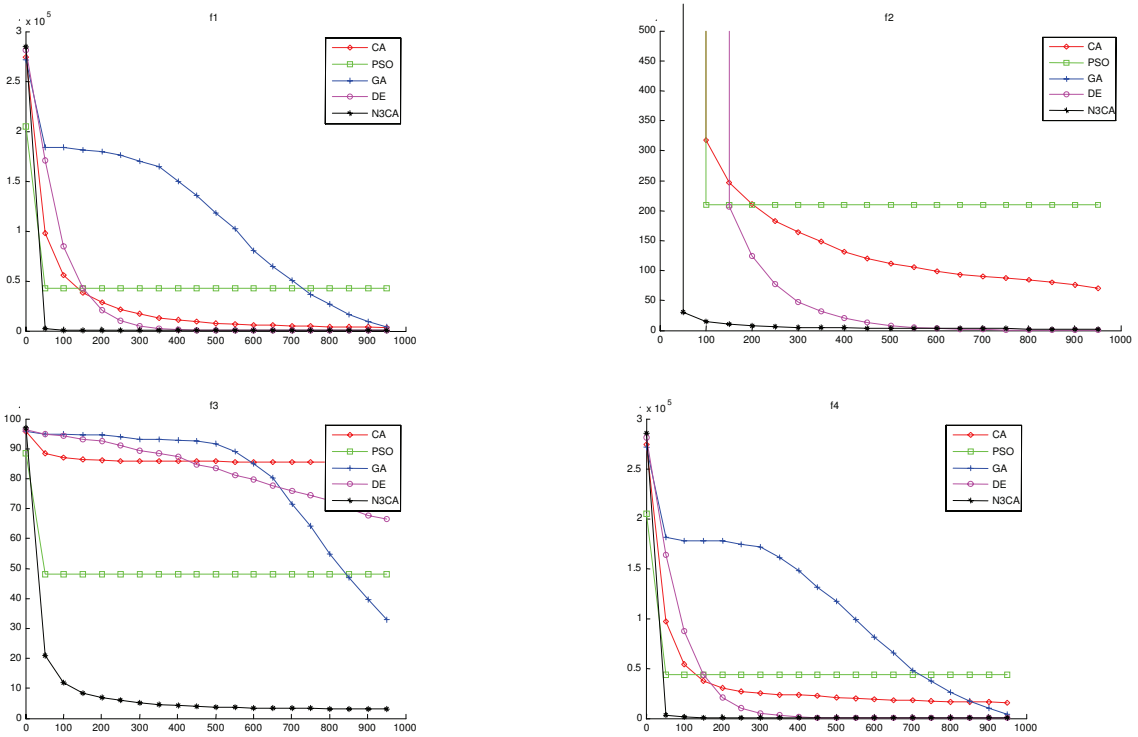

Fig. 3. Convergence performance of the five algorithms on the 10 test functions. $f1, f2, f3, f4$.

5. Conclusion

In this study, a novel version of CA is introduced and tested. The proposed novel version of CA benefits from a RBFNN in its normative knowledge source and hence is called N^3KCA. This version of CA uses two knowledge sources of normative and situational. The use of RBFNN in the normative knowledge source of CA makes it possible to update the normative knowledge by means of experiences which are obtained during the optimization process and the learning capability of RBFNN. The proposed algorithm pretends what the human brain does, i.e. to update the new values of the bounds of its normative knowledge, using the previous values of its normative knowledge and its experiences gained during the optimization. In order to optimize the values of the RBFNN used in the N^3KCA, evolutionary methods are utilized. A novel mutation operator is used to train RBFNN. The novel mutation operator is less complex and easy to implement. In addition, the output weights of the RBFNN are trained using PSO.

In order to update the RBFNN and consequently the normative knowledge source, the input of the RBFNN is selected by means of an acceptance function which benefits from a discrete reproducing algorithm. The lower and upper bounds of normative knowledge source are obtained using the updated RBFNN and the selected individual which is called *RAI*. The new population is generated by means of the influence function in the form of two mutation operators introduced by normative and situational knowledge sources. In order to show the efficacy of the proposed algorithm, it is compared with four other well known optimization methods namely GA, PSO, DE and existing version of CA using several unimodal and multimodal benchmark optimization problems.

In order to have a fair comparison, the comparisons are done in two cases: same number of population size and same order of complexities. The obtained results show that the N^3KCA can successfully jump out of the local minima on all of the multimodal functions and surpasses all the other algorithms when the dimension of the problem is high. In addition, the convergence graphs of the algorithms depict that the proposed method is capable of obtaining high quality solutions much faster than other mentioned algorithms.

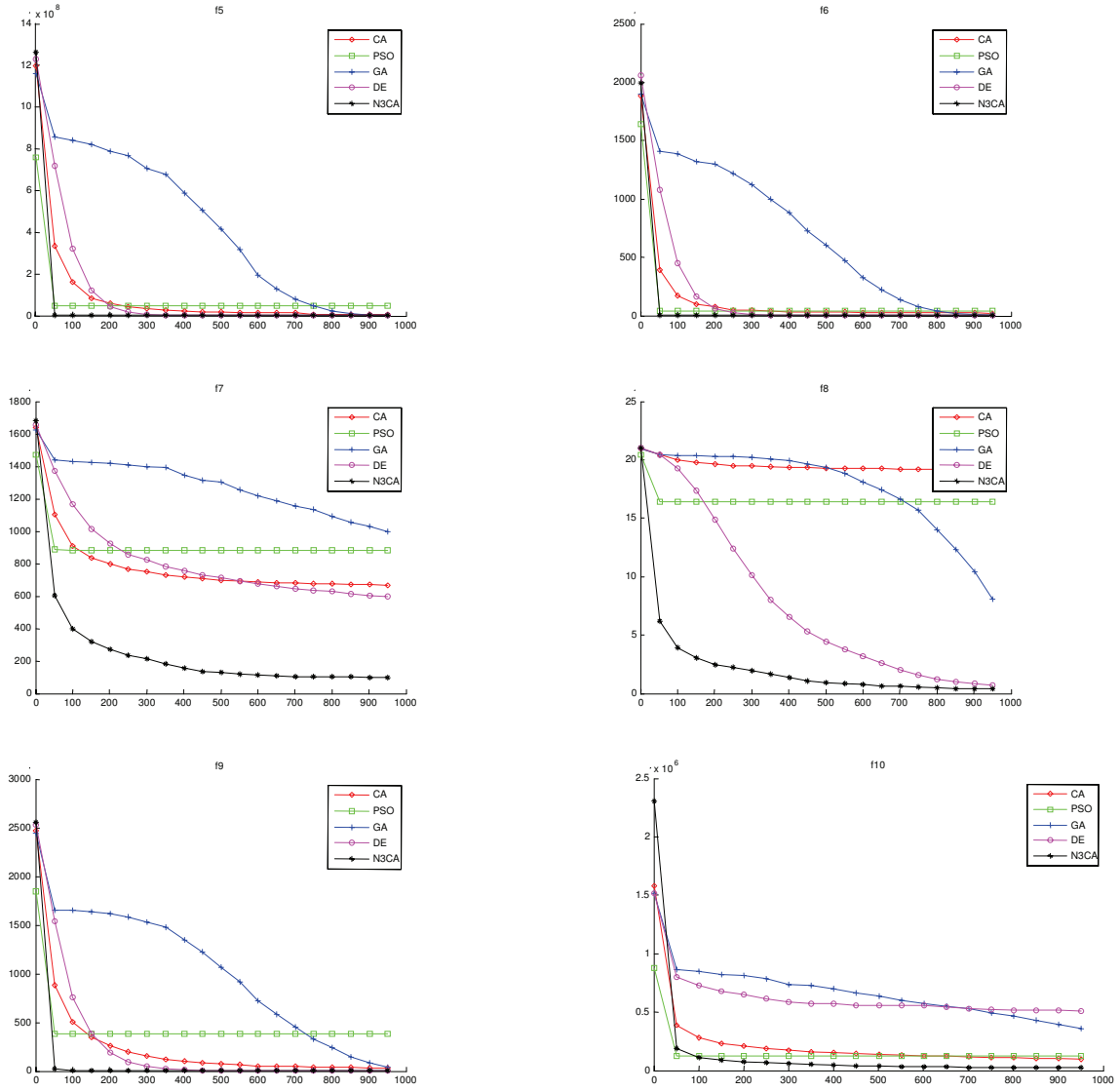

Fig. 4. (*Continued.*) Convergence performance of the five algorithms on the 10 test functions. $f_5, f_6, f_7, f_8, f_9, f_{10}$.

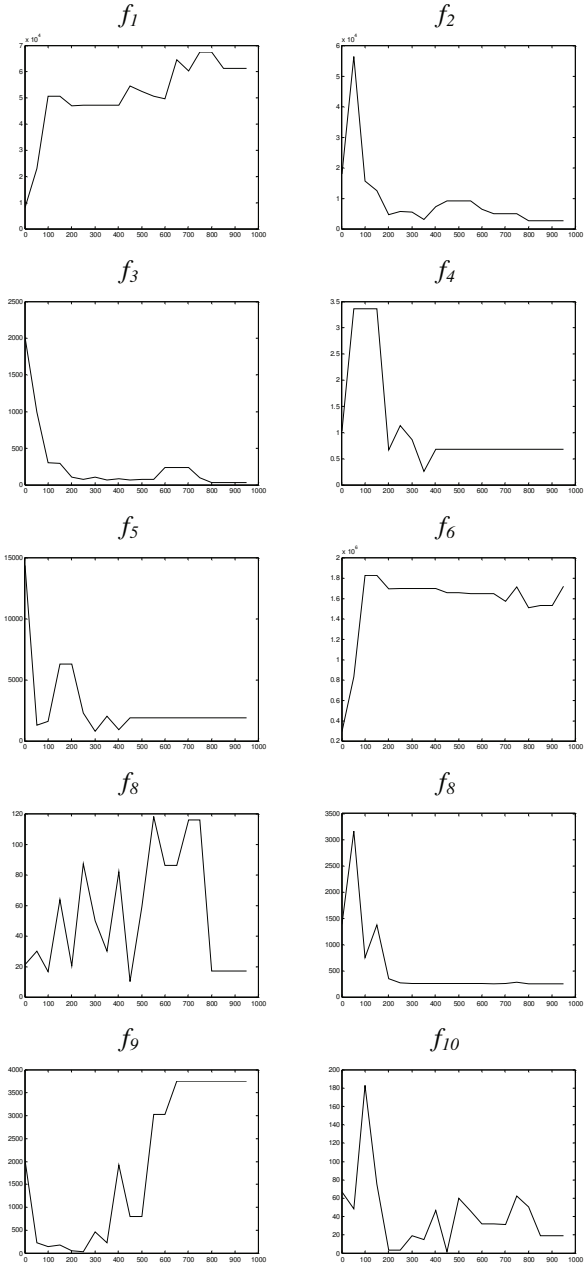

Fig 5. Frobinous norm of the weights of the normal RBFNN used in the normative knowledge source of the proposed algorithm.

6. References

[1] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning". Reading, MA: Addison-Wesley, 1989.

[2] Fogel, D. B. "Evolving Artificial Intelligence. " Ph. D. Thesis, San Diego, CA: University of California.1992

[3] J. Kennedy and R.C. Eberhart, "Particle swarm optimization," in Proceeding of the IEEE International Conference on Neural Networks, Perth, Australia, IEEE Service Center, pp. 12–13, 1995.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 26, no. 1, pp 29–41, 1996.

[5] R.G. Reynolds, "An introduction to cultural algorithms," in Proceedings of the 3rd Annual Conference on Evolutionary Programming, World Scientific Publishing, pp 131–139, 1994.

[6] R.G. Reynolds, "On modeling the evolution of Hunter-Gatherer decision-making systems," in Geographical Analysis, vol. 10, no. 1, pp 31–46, 1978.

[7] C. Chung and R.G. Reynolds, "CAEP: An evolution-based tool for real-valued function optimization using cultural algorithms," in International Journal on Artificial Intelligence Tools, vol. 7, no. 3, pp. 239–291, 1998.

[8] L.S Coelho, P. Alotto,"Electromagnetic Optimization Using a Cultural Self-Organizing Migrating Algorithm Approach Based on Normative Knowledge", in IEEE Transactions on Magnetics, vol. 45, no. 3, pp 1446-1449, 2009.

[9] R.G. Reynolds, M. Ali, "computing with social fabric The Evolution of Social Intelligence within a Cultural Framework",**I**n IEEE computational intelligence magazine, 2008

[10] C.J. Lin, C.H. Chen, C.T. Lin,"A Hybrid of Cooperative Particle Swarm Optimization and Cultural Algorithm for Neural Fuzzy Networks and Its Prediction Applications", in IEEE Transactions on Systems, Man, and Cybernetics —part C, vol. 39, no. 1, pp 55-68,2009

[11] R. G. Reynolds, B. Peng, "Cultural Algorithms: Modeling of How Cultures Learn to Solve Problems" ,in Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2004

[12] R.G. Reynolds, "An adaptive computer model of plant collection and early agriculture in the eastern valley of Oaxaca," in *Guila Naquitz: Archaic Foraging and Early Agriculture in Oaxaca*, Mexico, K. V. Flannery Ed, Academic Press, pp. 439–500.

[13] J. Adams and S. Payandeh, "Methods for low-velocity friction compensation:Theory and experimental study," *J. Robot. Syst.*, vol. 13, no.6, pp. 391–404, 1996.

[14] S. Chen and S. A. Billings, "Neural network for nonlinear dynamic system modelling and identification," *Int. J. Contr.*, vol. 56, pp.

[15] J. Gonzalez, I. Rojas, J. Ortega, H. Pomares, F. J. Fernandez, and A. F.Diaz, "Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation,"*IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1478–1495,Nov. 2003.

[16] M. J. Er, W. Chen, and S. Wu, "High-speed face recognition based on discrete cosine transform and RBF neural networks," *IEEE Trans.Neural Netw.*, vol. 16, no. 3, pp. 679–691, May 2005.

[17] S. S. Ge, C. C. Hang, T. H. Lee, and T. Zhang, *Stable Adaptive Neural Network Control*. Boston, MA: Kluwer, 2001.

[18] X. Hong, C. J. Harris, S. Chen, and P. M. Sharkey, "Robust

Co-published by Atlantis Press and Taylor & Francis

nonlinear model identification methods using forward regression," *IEEE Trans.Syst., Man, Cybern. A, Syst., Humans*, vol. 33, no. 4, pp. 514–523, Jul.2003.

[19] D. S. Huang, "The united adaptive learning algorithm for the link weights and the shape parameters in rbfn for pattern recognition," *Int.J. Pattern Recognit. Artif. Intell.*, vol. 11, no. 6, pp. 873–888, 1997.

[20] K. Li, J. Peng, and E. W. Bai, "A two-stage algorithm for identification of nonlinear dynamic systems," *Automatica*, vol. 42, no. 7, pp.1189–1197, 2006.

[21] Y. Li, S. Qiang, X. Zhuang, and O. Kaynak, "Robust and adaptive back stepping control for nonlinear systems using RBF neural networks," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 693–701, Sep. 2004.

[22] Y. J. Oyang, S. C. Hwang, Y. Y. Ou, C. Y. Chen, and Z. W. Chen, "Data classification with radial basis function networks based on a novel kernel density estimation algorithm," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 225–236, Jan. 2005.

[23] M. M. Polycarpou, "Fault accommodation of a class of multivariable nonlinear dynamical systems using a learning approach," *IEEE Trans. Autom. Control*, vol. 46, no. 5, pp. 736–742, May 2001.

[24] R.Iacoban, R. G. Reynolds, J. Brewste, "Cultural swarms: modeling the impact of culture on social interaction and problem solving," Proc IEEE Swarm Intelligence Symposiu, Indianapolis,MI, USA, PP. 205-211,2003

[25] R.G. Reynolds and S.M. Saleem, "The impact of environmental dynamics on cultural emergence," in *Perspectives on Adaptions in Natural and Artificial Systems*. Oxford University Press, pp. 253–280, 2001.

[26] X. Jin and R.G. Reynolds, "Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach," in *Proceeding of the 1999 Congress on Evolutionary Computation*, pp. 1672–1678, 1999.

[27] C.chung, "knowledge-based approaches to self-adaptation in cultural algorithms", Ph. D. thesis, Wayne State University, 1997.

[28] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

29] X. Yao, Y. Liu, and G. M. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[30] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2005, pp. 1–50.

[31] G. Leng, T.M. McGinnity, G. Prasad, "Design for self-organizing fuzzy neural networks based on genetic algorithms", in IEEE Trans. Fuzzy Systems 14 (2006) 755–766.

[32] Vahid Seydi Gh.,M. Sh Aliyari, A. Sharifi, M. Teshnehlab , "Multi Objective Optimization of ANFIS Structure",in International Conference on Intelligent and Advanced Systems 2008.

[33] H. Surmann, "Genetic optimization of a fuzzy system for charging batteries",in IEEE Trans. Ind. Electron. 43 (1996) 541–548.

[34] A. Bastian, "Identifying fuzzy models utilizing genetic programming", in Fuzzy Sets and Systems 113 (2000) 335–350.

[35] Y. Chen, B. Yang, A. Abraham, L. Peng, "Automatic design of hierarchical Takagi–Sugeno type fuzzy systems using evolutionary algorithms", in IEEE Trans. Fuzzy Systems 15 (3) (2007).

[36] Y. Shi, R. Eberhart, Y. Chen, "Implementation of evolutionary fuzzy systems", in IEEE Trans. Fuzzy Systems 7 (2) (1999).

[37] T.P.Wu, S.M. Chen, "A new method for constructing membership functions and fuzzy rules from training examples", in IEEE Trans. Systems Man Cybernet. Part B Cybernet. 29 (1) (1999).

[38] Vahid Seydi Gh.,M. Sh Aliyari, M. Teshnehlab , "Training ANFIS Structure with Modified PSO Algorithm", in Meditranean Conference on Control and Automation, 2007.

[39] M.Sh. Aliyari, M. Teshnehlab, A.K. Sedigh, "A novel training algorithm in ANFIS structure", in American Control Conf., 2006.

[40] M. Sh. Aliyari, M. Teshnehlab, A. K. Sedigh, "A novel hybrid learning algorithm for tuning ANFIS parameters using adaptive weighted PSO",in IEEE Internat.FuzzySystemConf.,2007.

[41] M. Sh. Aliyari, M. Teshnehlab, A. K. Sedigh, "Identification using ANFIS with intelligent hybrid stable learning algorithm approaches" ,in Neural Comput. Appl. (2008).

[42] G. Onwubolu and B. Babu, "New Optimization Techniques in Engineering, " Berlin, Germany: Springer-Verlag, 2004.

[43] R. Eberhart, Y. Shi, and J. Kennedy, "Swarm Intelligence, " San Mateo,CA: Morgan Kaufmann, 2001.

[44] M. Clerc, "Particle Swarm Optimization. Amsterdam,"The Netherlands:ISTE Publishing, 2006.

[45] T. Back, "Evolutionary Algorithms in Theory and Practice," Oxford,U.K.: Oxford Univ. Press, 1996.

[46] Z. Michalewicz, "Genetic Algorithms +Data Structures = Evolution Programs," New York: Springer, 1992.

[47] D. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine," *Learning*. Reading, MA: Addison-Wesley, 1989.

[48] K. Price and R. Storn, "Differential evolution," *Dr. Dobb's Journal*, vol. 22, pp. 18–20, 22, 24, 78, Apr. 1997.

[49] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 22–34, Apr. 1999.