2008 IEEE Region 10 Colloquium and the Third International Conference on Industrial and Information Systems, Kharagpur, INDIA December 8 -10, 2008.
353

Memetic Differential Evolution Trained Neural Networks For Nonlinear System Identification

Bidyadhar Subudhi, Debashisha Jena
Department of Electrical Engineering,
National Institute of Technology,
Rourkela, India
(bidyadhar@nitrkl.ac.in)

Madan M. Gupta
Intelligent Systems Research Laboratory
University of Saskatchewan line
Saskatoon, Canada
Madan.Gupta@usask.ca

Abstract— **Several gradient-based approaches such as back-propagation, conjugate gradient and Levenberg Marquardt (LM) methods have been developed for training the neural network (NN) based systems. Still, in some situations, like multimodal cost function, these procedures may lead to some local minima, therefore, the evolutionary algorithms (EAs) based procedures were considered as a promising alternative. In this paper we focus on a memetic algorithm based approach for training the neural networks. We use the memetic algorithm (MA) for training the multilayer perceptrons as applied to nonlinear system identification. The proposed memetic algorithm is an alternative to gradient search methods, such as back-propagation, which have inherent limitations of many local optima. Here we have studied the identification of a nonlinear system using five different algorithms namely Back-propagation (BP), Genetic Algorithm (GA), Differential Evolution (DE), Genetic Algorithm Back-propagation (GABP), along with the proposed Differential Evolution plus Back-propagation (DEBP) approaches. In the proposed system identification scheme, we have exploited two global search methods namely genetic algorithm (GA), and differential evolution (DE). These two global search methods have been hybridized with the gradient descent method i.e. the back propagation (BP) algorithm. The local search BP algorithm is used as an operator for genetic algorithm and differential evolution. These algorithms have been tested on a standard benchmark problem for nonlinear system identification to prove their efficacy.**
Keywords- Differential evolution, Evolutionary computation, Nonlinear system identification, Back propagation

I. INTRODUCTION

SYSTEM identification using neural networks has been considered as promising approach due to its function approximation properties [1, 2] and performance in modeling and nonlinear system dynamics. The major disadvantage to the NN trained with EAs is that it is computationally expensive, as the evolutionary approach is normally slow. To overcome the slow convergence of the evolutionary approach to NN training, other gradient methods are combined to speed up the convergence by augmenting evolutionary algorithms with a local search technique.

Evolving Neural Networks (NNs) [3] have prompted a great deal of research into the use of Memetic Algorithms (MAs) to evolve the design of NNs. MAs[4, 5] are a class of stochastic global search heuristics in which Evolutionary Algorithm approaches are combined with problem-specific solvers such as BP and LM etc. The later methods can be applied either in isolation or in combination with EAs for handling optimization problems. But when used alone there may be problem for getting trapped in local minima. The hybridization is meant to either accelerate the discovery of good solutions, for which evolution alone would take too long to discover, or to reach solutions that would otherwise be unreachable by evolution or a local method alone. It is assumed that the evolutionary search provides for a wide exploration of the search space while the local search can somehow zoom-in on the basin of attraction of promising solutions.

A variant of evolutionary computing namely the Differential Evolution [6] is a population based stochastic optimization method similar to genetic algorithm [7] that finds an increasing interest in the recent year as an optimization technique in the identification of nonlinear systems due to its achievement of a global minimum. However, a little work has been reported on memetic differential evolution learning of neural network. Therefore, it attracts the attention of the present work for neural network training. In this work, a differential evolution hybridized with back propagation has been applied as a optimization method for feed-forward neural network. Differential Evolution (DE) is an effective, efficient and robust optimization method capable of handling nonlinear and multimodal objective functions.
A nonlinear system as considered in [8] has been chosen in this work for demonstrating the efficacy of the proposed hybrid evolutionary system identification. In this work, genetic algorithm and differential evolution are acting as global search methods which are individually combined with BP for training a feed-forward neural network. In the proposed

scheme, in each generation back-propagation acts as an operator which is applied after crossover and mutation operator.

The main contributions of the paper are as follows:

- The paper proposed a new training paradigm of neural networks combining an evolutionary algorithm i.e. GA and DE with a local search algorithm i.e. BP for getting faster convergence in comparison to only evolutionary computation and to avoid the possibility of the search process being trapped in local minima which is the greatest disadvantage of local search optimization.
- BP has been integrated as an operator in global searches for optimizing the weights of the neural network training enabling faster convergence of the EANN employed for nonlinear system identification.
- The identification performance of the proposed scheme has been compared with the EANN approach and BPNN approach to nonlinear system identification and found to be better than the above mentioned methods.

II. A BRIEF REVIEW ON DIFFERENTIAL EVOLUTION STRATEGY

In a population of potential solutions to an optimization problem within an n-dimensional search space, a fixed number of vectors are randomly initialized, and then new populations are evolved over time to explore the search space and locate the minima of the objective function. Differential evolutionary strategy (DES) uses a greedy and less stochastic approach in problem solving rather than the other evolutionary algorithms. DE combines simple arithmetical operators with the classical operators of recombination, mutation and selection to evolve from a randomly generated starting population to a final solution. The fundamental idea behind DE is a scheme whereby it generates the trial parameter vectors. In each step, the DE mutates vectors by adding weighted, random vector differentials to them. If the fitness of the trial vector is better than that of the target, the target vector is replaced by the trial vector in the next generation. Now we explain the working steps involved in employing a DE cycle.

Step 1: Parameter setup

Choose the parameters of population size, the boundary constraints of optimization variables, the mutation factor *(F)*, the crossover rate *(C)*, and the stopping criterion of the maximum number of generations *(g_{max})*.

Step 2: Initialization of the population

Set generation *g=0*. Initialize a population of *i*=1, *P* individuals (real-valued *d*-dimensional solution vectors) with random values generated according to a uniform probability distribution in the *d* dimensional problem space. These initial values are chosen randomly within user's defined bounds.

Step 3: Evaluation of the population

Evaluate the fitness value of each individual of the population. If the fitness satisfies predefined criteria save the result and stop, otherwise go to step 4.

Step 4: Mutation operation (or differential operation)

Mutation is an operation that adds a vector differential to a population vector of individuals. For each target vector $x_{i,g}$ a mutant vector is produced using the following relation,

$$v_{i,g} = x_{r_1,g} + F\,(x_{r_2,g} - x_{r_3,g}) \tag{1}$$

In Eqn. (1), F is the mutation factor, which provides the amplification to the difference between two individuals $(x_{r_2,g} - x_{r_3,g})$ so as to avoid search stagnation and it is usually taken in the range of [0, 1]. where $i, r_1, r_2, r_3 \in \{1, 2, ...P\}$ are randomly chosen numbers but they must be different from each other. P is the number of population.

Step 5: Recombination operation

Following the mutation operation, recombination is applied to the population. Recombination is employed to generate a trial vector by replacing certain parameters of the target vector with the corresponding parameters of a randomly generated donor (mutant) vector. There are two methods of recombination in DE, namely, binomial recombination and exponential recombination.

In binomial recombination, a series of binomial experiments are conducted to determine which parent contributes which parameter to the offspring. Each experiment is mediated by a crossover constant, *C*, (0 ≤ *C* ≤1). Starting at a randomly selected parameter, the source of each parameter is determined by comparing *C* to a uniformly distributed random number from the interval [0, 1]. If the random number is greater than *C*, the offspring gets its parameter from the target individual; otherwise, the parameter comes from the mutant individual. In exponential recombination, a single contiguous block of parameters of random size and location is copied from the mutant individual to a copy of the target individual to produce an offspring. A vector of solutions are selected randomly from the mutant individuals when $rand_j$ ($rand_j \in [0,1]$, is a random number) is less than *C*.

$$t_{j,i,g} = \begin{cases} v_{j,i,g} & if\ (rand_j \le C)\ or\ j = j_{rand} \\ x_{j,i,g} & otherwise \end{cases} \tag{2}$$

$j = 1, 2 \ldots d$, where d is the number of parameters to be optimized.

Step 6: Selection operation

Selection is the procedure of producing better offspring. If the trial vector $t_{i,g}$ has an equal or lower value than that of its target vector, $x_{i,g}$ it replaces the target vector in the next generation; otherwise the target retains its place in the population for at least one more generation.

$$x_{i,g+1} = \begin{cases} t_{i,g}, & if\ f(t_{i,g}) \le f(x_{i,g}) \\ x_{i,g}, & otherwise \end{cases} \tag{3}$$

Once new population is installed, the process of mutation, recombination and selection is replaced until the optimum is located, or a specified termination criterion is satisfied, e.g., the number of generations reaches a preset maximum $g_{\max}$.

At each generation, new vectors are generated by the combination of vectors randomly chosen from the current population (mutation). The upcoming vectors are then mixed with a predetermined target vector. This operation is called recombination and produces the trial vector. Finally, the trial vector is accepted for the next generation if it yields a reduction in the value of the objective function. This last operator is referred to as a selection.

III. MEMETIC ALGORITHM

Memetic algorithms are hybrid algorithms that combine both the evolutionary algorithms and local search techniques. For different classes of optimization problems, memetic algorithms have been proved to be much faster and more accurate than the evolutionary algorithms.

To describe this mechanism with more details, we will introduce the following formal framework. Let us first introduce the population cardinality P and the population at the g^th generation, $\Pi_g = \{\underline{c}_{1,g} \dots \underline{c}_{P,g}\}$.

Let H be the operator may be mutation/crossover/reproduction defined as

$$H \mid \Pi_g \in \mathbb{R}^{d*P} \to \underline{x}(x_1 \dots x_P) \in \mathbb{R}^P$$

which associate to each population the fitness vector of its elements.

Let *R, M be* the recombination and the mutation operators respectively. These operators are called the reproduction operators as well and are defined as

$$R \mid \Pi \in \mathbb{R}^d \times \mathbb{R}^P \to \Pi \in \mathbb{R}^d \times \mathbb{R}^P$$

$$M \mid \Pi' \in \mathbb{R}^d \times \mathbb{R}^P \to \Pi'' \in \mathbb{R}^d \times \mathbb{R}^P$$

Let us denote with *LS* the local search operator, i.e., the operator which produces a new population by applying the *LS* with starting points equal to the individuals in the current population:

$$LS \mid \Pi'' \in \mathbb{R}^d \times \mathbb{R}^P \to \Pi''' \in \mathbb{R}^d \times \mathbb{R}^P$$

where n is the number of parameters and k is the number of population. Then applying the selection operator the next generation population can be determined.

$$\Pi_{g+1} = selection\left\{R\left(M\left[LS\left(\Pi_g\right)\right]\right)\right\}$$

In a Hybrid Evolutionary Algorithm, the role of the Evolutionary Algorithm is essentially to explore the searching space and locate the more promising regions.

IV. PROPOSED DIFFERENTIAL EVOLUTION BACK-PROPAGATION TRAINING ALGORITHM FOR NON-LINEAR SYSTEM IDENTIFICATION

In the sequel, we describe how a memetic differential evolution (DE) is applied for training neural network in the frame work of system identification (see Algorithm-1). DE can be applied to global searches within the weight space of a typical feed-forward neural network. Output of a feed-forward neural network is a function of synaptic weights $\mathbf{w}$ and input values $\mathbf{x}$, i.e. $\hat{\mathbf{y}} = f(\mathbf{x}, \mathbf{w})$.The role of BP in the proposed algorithm has been described in section I. In the training process, both the input vector $\mathbf{x}$ and the output vector $\mathbf{y}$ are known and the synaptic weights in $\mathbf{w}$ are adapted to obtain appropriate functional mappings from the input $\mathbf{x}$ to the output $\mathbf{y}$. Generally, the adaptation can be carried out by minimizing the network error function $\mathbf{E}$ which is of the form $\mathbf{E}(\mathbf{y}, f(\mathbf{x}, \mathbf{w}))$. In this work we have taken $\mathbf{E}$ as mean squared error i.e. $\mathbf{E} = \frac{1}{N}\sum_{k=1}^{N}[\mathbf{y} - f(\mathbf{x}, \mathbf{w})]^2$, where N is the number of data considered. The optimization goal is to minimize the objective function $\mathbf{E}$ by optimizing the values of the network weights, $\mathbf{w} = (w_1, \dots, w_d)$.

ALGORITHM-1: Differential Evolution Back-Propagation (DEBP) Identification Algorithm:

Step 1.

Initialize population **pop**: Create a population from randomly chosen object vectors with dimension P ×d, where P is the number of population and d is the number of weights of the neural network

$$\mathbf{P}_g = (\mathbf{w}_{1,g}, \dots, \mathbf{w}_{P,g})^T, \ g = 1, \dots, g_{\max}$$

$$\mathbf{w}_{i,g} = (w_{1,i,g}, \dots, w_{d,i,g}), \quad i = 1, \dots, P$$

where d is the number of weights in the weight vector and in $\mathbf{w}_{i,g}$, i is index to the population and g is the iteration (generation) to which the population belongs.

Step 2.

Evaluate all the candidate solutions inside the **pop** for a specified number of iterations.

Step 3.

For each i^{th} candidate in **pop,** select the random population members, $r_1, r_2, r_3 \in \{1, 2, \dots P\}$

Step 4.

Apply a mutation operator to each candidate in a population to yield a mutant vector i.e.

$$v_{j,i,g+1} = w_{j,r1,g} + F(w_{j,r2,g} - w_{j,r3,g}),\ for\ j = 1,...d$$
$$(i \neq r_1 \neq r_2 \neq r_3) \in \{1,...P\}\ and\ F \in [0,1]$$

where "*F*" denotes the mutation factor.

Step 5.

Apply crossover i.e. each vector in the current population is recombined with a mutant vector to produce trial vector.

$$t_{j,i,g+1} = \begin{cases} v_{j,i,g+1} & if\ rand_j[0,1) \le C \\ w_{j,i,g} & otherwise \end{cases}$$

where $C \in [0,1]$

Step 6.

Apply Local Search back propagation algorithm i.e. each trial vector will produce a lst-trial vector

$$lst_{j,i,g+1} = bp\left(t_{j,i,g+1}\right)$$

Step 7.

Apply selection i.e. between the local search trial (lst-trial) vector and the target vector. If the lst-trial vector has an equal or lower objective function value than that of its target vector, it replaces the target vector in the next generation; otherwise, the target retains its place in the population for at least one more generation

$$\mathbf{w}_{i,g+1} = \begin{cases} lst_{i,g+1}, & if\ \mathbf{E}(\mathbf{y}, f(\mathbf{x}, \mathbf{w}_{i,g+1})) \le \mathbf{E}(\mathbf{y}, f(\mathbf{x}, \mathbf{w}_{i,g})) \\ \mathbf{w}_{i,g}, & otherwise \end{cases}$$

Step 8.

Repeat steps 1 to 7 until stopping criteria is reached

V. RESULTS AND DISCUSSION

In this section we present the performance of the proposed Differential Evolution Back-Propagation (DEBP) Identification Algorithm using the simulation studies on a bench mark problem for the identification of a nonlinear discrete system [8] expressed by

$$y_p(k+1) = \frac{y_p(k)[y_p(k-1)+2][y_p(k)+2.5]}{8.5 + [y_p(k)]^2 + [y_p(k-1)]^2} + u(k) \quad (4)$$

Where $y_p(k)$ is the output of the system at the kth time step and *u(k)* is the plant input which is uniformly bounded function of time. The plant is stable for *u (k)* $\in$ [-2 2].

For the identification of the plant described in Eqn. (4), let the neural model be in the form of

$$\hat{y}(k+1) = f(y_p(k), y_p(k-1)) + u(k) \quad (5)$$

where $f(y_p(k), y_p(k-1))$ is the nonlinear function of $y_p(k)$ and $y_p(k-1)$. The inputs to the neural network are $y_p(k)$ and $y_p(k-1)$ .The output from neural network is $\hat{y}(k+1)$. The goal is to train the neural network such that when an input *u (k)* is simultaneously presented to the to the nonlinear system (13) as well as to neural network (14), the neural network outputs $\hat{y}(k)$ will finally approach the nonlinear system output $y_p(k)$ as close as possible.

For plant identification, the morphology of the neural network consisted of twenty one numbers of neurons in the hidden layer. After 100 epochs the training of the neural identifier has been stopped. During training period, input *u(k)* was a random white noise signal, but after the training is over, its prediction capability were tested for input

$$u(k) = 2\cos(2\pi k/100) \quad k \le 200 \quad (6a)$$

$$u(k) = 1.2\sin(2\pi k/20) \quad 200 < k \le 500 \quad (6b)$$

A. Identification Using Back-Propagation Algorithm (BP) (Figures 1, 2)

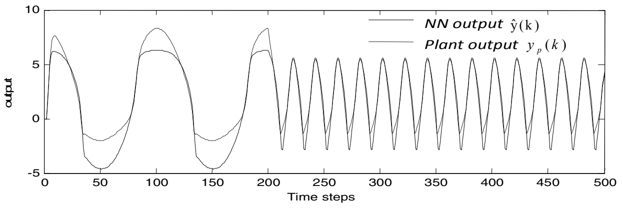

Fig. 1 BP identification

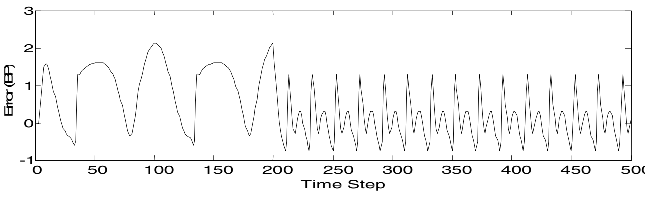

Fig.2 Error in modeling (BP identification)

Figure 2 shows the system identification results obtained using the back propagation algorithm for training the feed-forward neural network. Figure 3 shows the identification error.

B. Identification Using Genetic Algorithm (GA) (Figures 3, 4)

Figure 3 shows the identification performance of the system using genetic algorithm as learning algorithm for the given neural network. The same neural network configuration i.e. twenty one number of neurons are taken into account. After 100 epochs it was found that the squared error is more than conventional back propagation also taking more time to converge. Fig.4 shows the error between actual and GA identification.

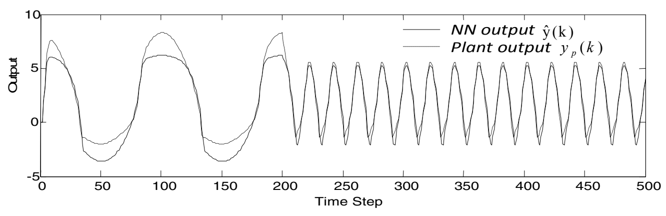

Fig. 3 GA identification.

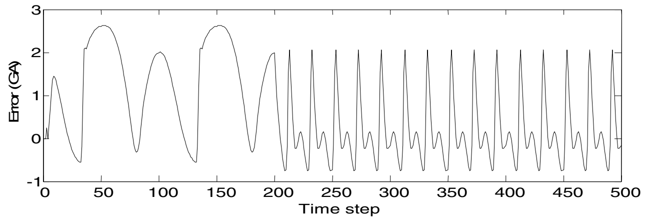

Fig.4 Error in modeling (GA identification)

C. Genetic algorithm Back-propagation (GABP) Identification (Figures 5, 6)

Figure 5 shows the comparison between GABP identification performance and actual output. The identification error between the actual output and the GABP output is shown in Fig.6

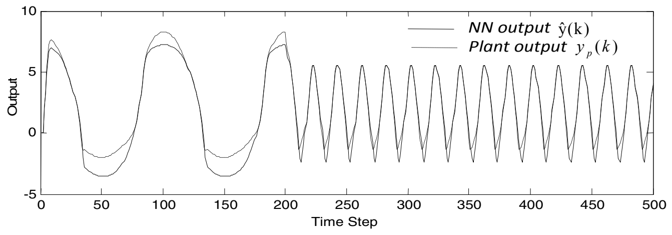

Fig. 5 GABP identification.

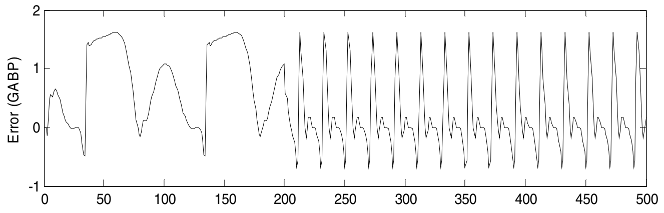

Fig. 6 Error in modeling (GABP identification)

D. Differential Evolution (DE) Identification (Figures 7, 8)

Figure 7 shows the identification performance between the differential evolution and the actual output. It was found that the performance is better than GA but worst than GABP. The identification error between the actual output and the DE output is shown in Fig.8.

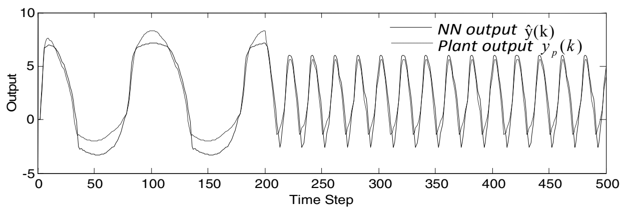

Fig. 7 DE identification

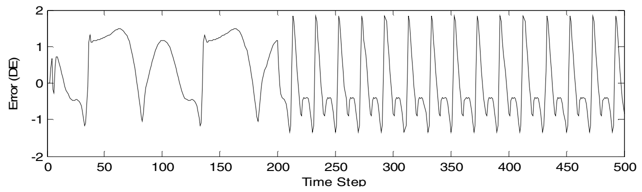

Fig. 8 Error in modeling (DE identification)

E. Differential Evolution plus the Back-propagation (DEBP) Identification (Figure 9, 10)

From Fig.9 it is clear that the proposed method i.e. DEBP identification is more effective than other mentioned approaches as per as identification performance and speed of convergence is concerned. Figure 10 shows the error between the proposed one and the actual one

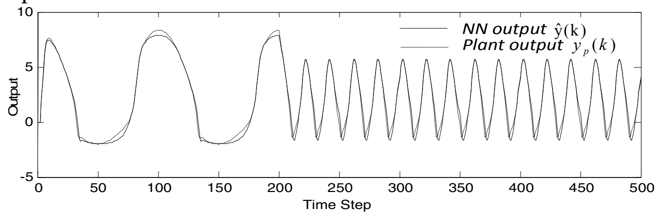

Fig. 9 DEBP identification

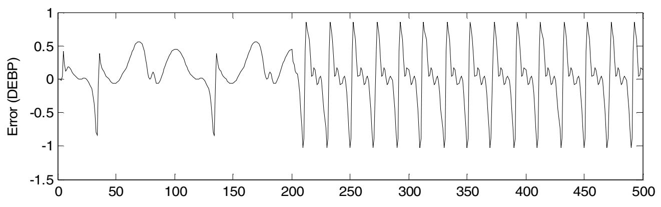

Fig. 10 Error in modeling (DEBP identification)

F. Performance Comparison of all the five identification methods cited in this paper

Figure 11 depicts the mean square error (MSE) profiles for all the five different identification methods (BP, GA, GABP, DE and DEBP). In these five methods, we have proposed a new identification scheme, namely the Differential Evolution plus the Back-propagation (DEBP) identification approach. From this figure it is clear that the SSE with the proposed method DEBP converges to zero very fast taking only about 20th iteration while the error curves with the other system identification methods (BP, GA,GABP, and DE) converges to zero taking over 100th iterations. Hence it is important to note that the proposed DEBP system identification exhibits better convergence characteristics. All the simulations have been performed in MATLAB using same set of parameters i.e. population size, number of generations, upper and lower bounds of weights and number of hidden layer neurons.

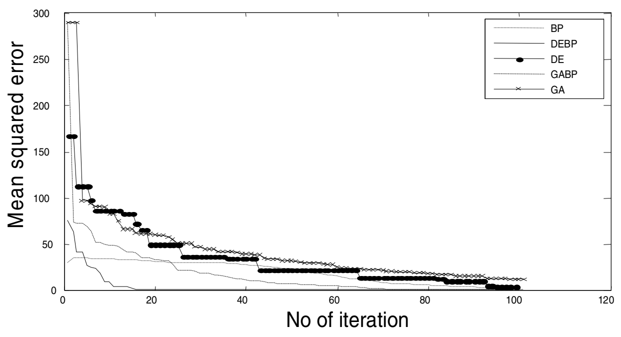

Figure 11 comparisons on the convergence of the mean squared error (MSE) for all the seven methods.

Table 1 Parameters used in Simulation Studies

Total sampling number, T	500
Population size, S	20
Number of iterations (generation)	100
Upper and lowerbound on weights	[-1 1]
BP learning parameter, η	0.55
Number of hidden layer neurons	21
Parameter for DE&DEBP Algorithm	
Mutation constant factor , *F*	0.6
Cross over constant, *C*	0.5
BP learning parameter, η	0.55
Parameter for GA&GABP Algorithm	
Mutation probability	0.002
Cross over constant Probability	1
BP learning parameter, η	0.55

Table 2 gives the comparison of performance of all the seven methods in terms of mean squared error (MSE). From the results it is clear that for a particular number of iteration i.e. 100, the proposed DEBP algorithm has a mean squared error (MSE) of 0.0625. It is found that out of all the five methods the memetic approaches i.e. GABP, and DEBP are having faster convergence in comparison to other local search and evolutionary computing approaches. Finally it is concluded that the proposed memetic DEBP is having better identification performance and faster convergence in comparison to memetic GABP algorithm which indicates DE is outperforming than its counterpart GA.

Table 2 Comparison of Performances of five methods

Metho	*Mean squared error(MSE)*	*Number of iteration at which the MSE converges approximately to zero*
BP	2.6086	>100
GA	11.4156	>100
GABP	0.2852	70
DE	3.9645	>100
DEBP	0.0625	20

VI. Conclusions

In this paper we provide an extensive study of memetic algorithms (MAs) applied to nonlinear system identification. From the results presented in this paper it has been found that the proposed DEBP memetic algorithm applied to neural network learning exhibits better result in terms of faster convergence and lowest sum squared error (SSE) amongst all the seven methods (i.e. BP, GA, GABP, DE. and DEBP). The proposed method DEBP exploits the advantages of both the local search and global search. It is interesting to note that the local search pursued after the mutation and crossover operation that helps in intensifying the region of search space which leads to faster convergence. We investigated the performance of the proposed version of the DEBP algorithm using a benchmark nonlinear system identification problem. The simulation studies showed that the proposed algorithm of DEBP outperforms in terms of convergence velocity among all the seven discussed algorithms. The overall performance of the DEBP scheme was better than the other approaches and the overall performance of the newly proposed DEBP algorithm was superior to other methods i.e. GABP. This shows the advantages of using DEBP over other evolutionary computation such as GA.

References

[1]. Narendra, K.S., and Parthaasarathy, K., "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans Neural Networks*, vol.1, pp 4-27, 1990.

[2]. Gupta M. M., Jin L., and Homma N, 2002. Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory, John Wiley and Sons, Inc., New York.

[3]. Yao, X. "Evolutionary artificial neural networks". *Int. Journal of Neural Systems*, vol.4, pp 203-222, 1993.

[4]. Merz, P., "Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies", PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 2000.

[5]. Petalas, Y.G., Parsopoulos, K.E., and Vrahatis, M.N., "Memetic particle swarm optimization", *Annals Operation Research,* DOI 10.1007/s10479-007-0224-y, vol.156, pp 99-127, 2007.

[6]. Storn, R., and Price, K., "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces". *Journal of Global Optimization*, vol.11, 341–359, 1997.

[7]. Vavak, F., Fogarty, T. and Jukes, K "A genetic algorithm with variable range of local search for tracking changing environments". In *Proceedings of the 4th Conference on Parallel Problem Solving from Nature*, 1996

[8]. C.-T. Lin, and C.S.George Lee, Neural *Fuzzy Systems: A Neuro-fuzzy Synergism to Intelligent Systems, Prentice Hall International, Inc.,* New Jersey: 1996.