**British Journal of Mathematics & Computer Science**
**19(1): 1-15, 2016; Article no.BJMCS.29060**
***ISSN: 2231-0851***

**SCIENCEDOMAIN** *international*
www.sciencedomain.org

# Online Elman Neural Network Training by Genetic Algorithm

**Ali Hussein Hasan[1*] and Watheq Hayawi Laith[1]**

*[1]College of Administration and Economic, University of Sumer, Al-Rifaei, Iraq.*

***Authors' contributions***

*This work was carried out in collaboration between both authors. Both authors read and approved the final manuscript.*

***Article Information***

DOI: 10.9734/BJMCS/2016/29060
Editor(s):
(1) Qiankun Song, Department of Mathematics, Chongqing Jiaotong University, China.
Reviewers:
(1) Stephen Akandwanaho, University of KwaZulu-Natal, South Africa.
(2) Radosław Jedynak, Kazimierz Pulaski University of Technology and Humanities, Poland.
(3) Shiahn-Wern Shyue, National Sun Yat-sen University, Kaohsiung, Taiwan.
Complete Peer review History: http://www.sciencedomain.org/review-history/16534

***Received: 20th August 2016***
***Accepted: 30th September 2016***
***Published: 13th October 2016***

***Original Research Article***

## Abstract

Although most offline and online training algorithms based on gradient search techniques like backpropagation algorithm and its modifications or on Kalman filter approaches, it has been shown that these techniques are severely limited in their ability to find global solutions, they converge slowly, get local minimization too easily and courses oscillation. Global search techniques have been identified as a potential solution to this problem, but they are limited to offline training because of the long time of convergence. The paper is focused on presenting of applying online genetic algorithm to train recurrent artificial neural networks. Here; improvement are made on the real coding genetic algorithm by introducing a reserve elite chromosome. The new approach is tested on the Elman network (which generally suffer from very long training time) for several types of dynamic system plants. The simulation results show that the proposed algorithm is able to train ENN with less training data set in corresponding to Kalman filter training algorithm.

*Keywords: Fast genetic algorithm; online neural network training; Elman neural network.*

**Corresponding author: E-mail: ali.alsabool@gmail.com;*

## 1 Introduction

Artificial Neural Networks (ANN’s), also known as connectionist models, are rapidly evolving facet of artificial intelligence (AI) [1]. In particular, the main idea is to reproduce the intelligence and the capability to learn from examples, simulating the brain neuronal structure on a calculator. ANN’s are interconnected networks of simple elements which interact with the objects of the real world in the same way as biological nervous system does. Because of the biological basis for the artificial neural networks, it is not surprising that many of the terms used in their study are borrowed from neurophysiology [1]. The processing units are neurons, nodes or processors; while the connections between these units are known as interconnected, synapses or weights. The pattern of the connections between the units determines the architecture of the network, which in the extremes, can be fully interconnected (recurrent neural networks) or connected in one direction only (feed forward neural networks).

Multilayers feedforward neural network based on series connection of neuron layers, each one composed by a set of neurons connected in parallel. The signals flow from the input layer through the hidden layer(s) to the output layer via uni-directional connections. There are one or more layers between the input and output layers.

Feedforward networks can be only used for dynamic relationship between input and output variable by including lagged values of input and output variables in the input layer. Recurrent Neural Network (RNN) allows for an internal feedback in the system. Internal feedback is a more successful way to account for dynamics in the model. It contains the entire history of inputs as well as outputs [2]. Recurrent neural networks are fundamentally different from feedforward architectures in the sense that they not only operate on an input space but also on an internal state [3].

ANN’s must be learned before it is used; and there are many learning algorithms that can be used to train ANN’s; backpropagation (with its modifications) is currently the mainstay of artificial neural networks learning. BP algorithm has high accuracy, but it has some disadvantages: it converges slowly, gets local minimization too easily and courses oscillation. Local minimization can be solved by adjusting the initial weights, while slow convergence and the oscillation are usually coursed by getting into local minimization in the later period of the network training [4]. Miao et al. [5] indicated that the BP solutions are usually forced to the local minimum due to the gradient descent algorithm used to get weights of connections. Engoziner et al. [6] presented that BP uses some variation of the gradient technique, which is essentially a local optimizing method and thus has some inevitable drawbacks, such as easily trapping into the local optimal and dissatisfying generalization capability.

Genetic algorithms (or simply GAs) are powerful and widely applicable stochastic search and optimization methods based on the concepts of natural selection and natural evolution. Genetic algorithms were first invented by John Holland in 1960s and were developed by Holland and his students and colleagues at the university of Michigan in the 1960s and the 1970s [7]. GA shows great promise in complex domains because it operates in an iterative improvement fashion. The search performed by it is probabilistically concentrated towards regions of the given data set that have been found to produce a good classification behavior. An overview about GAs and their implementation in various fields was given by Goldberg [8] or Michalewicz [9]. GA may be used to do one or more of the following when it's combined with neural networks: - 1. Weight training for supervised learning and reinforcement learning applications. 2. Select training data and 3. Finding neural network architecture.

Consequently, newly research tends to hybridize several artificial intelligence (AI) techniques to improve the performance. Neural networks and genetic algorithms demonstrate powerful problem solving ability. They are based on quite simple principles, but take advantage of their mathematical nature: Non-linear iteration [10]. Topchy et al. [11] proposed two algorithms to learn ANN weights; in the first proposed algorithm, learning process can be considered as evolutionary adaptation of network parameters to optimal internal representation of the information being processed. The second algorithm mainly focuses on direct combination of genetic algorithm and delta- rule for training two subsets of multilayered perceptron's parameters, i.e. hidden and output layer weights respectively. Sexton et al. [12] made a comparison between the backpropagation and genetic algorithms. Sexton et al. [13] examined two well known global search techniques, Simulated Annealing and the Genetic Algorithm, and compare their performance in neural network training. Schiffmann et al. [14] wrote a technical report about using genetic algorithms in neural networks fields. Topchy et al. [15] presented an RBFN training algorithm based on evolutionary programming and cooperative evolution. Gruau [16] combined neural network architecture optimization and weights learning in single algorithm. Schaffer et al. [17] made a good survey of various combinations of genetic algorithms and neural networks. Fogel [18] used evolutionary programming to create neural networks that are capable of playing Tic-Tac-Toe. Wies [19] focused on the intersection of neural networks and evolutionary learning and showed the basic aspects of combination of these learning paradigm. Gupta and Sexton [20] showed that the use of GA can provide better results for training feedforward NN than the traditional techniques of backpropagation. Zhang Lijun [21] optimized Elman network initial weights by GA algorithm, then applied the ENN to predict stock price and made an efficient GA-Elman neural network stock prediction model. Zhang Xiuling [22] established a model of Elman network prediction for nickel-metal hydride battery capacity, he used GA to optimize the initial and threshold. Wang Tian'e [23] used GA to optimize the structure and weights, of Elman network model and successfully applied it to Dongfeng Motor's stock price prediction. Other interesting papers deal with NN optimized by GA can be found in [24-33].

The paper is structured as follows. Sections 2 give an overview of Elman recurrent neural networks. Section 3 describes the classical genetic algorithms. Section 4 describes in details the proposed online genetic algorithm for training ENN. Section 5 the results of applying the proposed genetic algorithm for training ENN to model three plants. The conclusion is described in section 6.

## 2 Elman Recurrent Neural Networks

In 1990 Elman introduced a simple recurrent neural network that is mainly used to deal with sound processing. The Elman network can be considered as a forward network with a local memory unit and local feedback connections. In addition to the input unit, the hidden units, and the output unit, there are also context units. The hidden units can have linear or nonlinear activation functions. The context units are used to remember the previous activations of the hidden units and considered to function as one-step time delays. The block diagram of Elman network is shown in Fig. 1. The output of hidden associates with its input through the delay and storage of undertake layer. This way of association is sensitivity to historical data, and internal feedback network can increase the ability of handling dynamic information. Remembrances the internal state makes it to have the dynamic mapping function, which make the system to have the ability to adapt to time varying characteristics [34].

Let the input signal is ($u_k$);

$$S_i^h = W_{ch}X_{k-1} + W_{ih}u_k, \tag{1}$$

$$X_k = F(S_i^h), \tag{2}$$

$$S_h^o = W_{ho}X_k, \tag{3}$$

$$y_{k+1} = S_h^o, \tag{4}$$

where:-

$S_i^h$ is the net sum of the hidden layer neurons.
$X_k$ is the output of the hidden layer neurons.
$S_h^o$ is the net sum of the output layer neurons.

$y_{k+1}$ is the outputs of the network.
$W_{ih}$ is the connection weight between the input and hidden layer.
$W_{ch}$ is the connection weight between the hidden layer and context layer.
$W_{ho}$ is the connection weight between the hidden and output layer.
F activation function of the hidden layer neurons.

The activation function of the hidden layer  is tangential function as described in equation (5); while a linear type used for the of the output layer as described in equation (4).

$$F(S_i^h) = (e^{(Sih)} - e^{(-Sih)})/(e^{(Sih)} + e^{(-Sih)}) \quad (5)$$

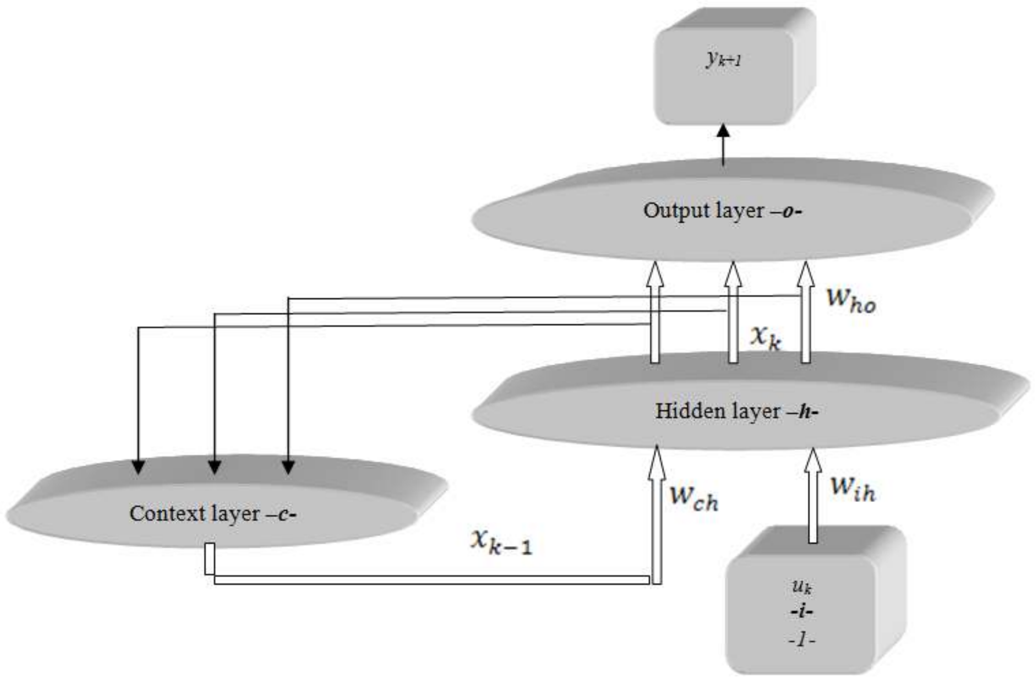

**Fig. 1. Elman neural network structure**

At the time k, the current input $u_k$ and the previous activations of the hidden units $x_{k-1}$ are used as inputs to the network. At this moment, the network acts as a feed-forward network and propagates these inputs forward to produce the output $y_{k+1}$.

## 3 Genetic Algorithm

GAs have been shown to be an effective strategy in the offline design of many fields. The GAs are not too demanding, as could be natural to expect, in terms of their needs of computational power, being applied so far to a wide range of problems, going from the fields of Combinatorial or Numeric Optimization to Image Processing or even Machine Learning. GAs operate on a population of potential solutions applying the principle of “survival of the fittest” to produce (hopefully) better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

A genetic algorithm has three major components [34]. The first component is related with the creation of an initial population of m randomly selected individuals. The initial population shapes the first generation. The second component inputs m individuals and gives as output an evaluation for each of them based on an objective function known as fitness function. This evaluation describes how close to our demands each one of these m individuals is. Finally the third component is responsible for the formulation of the next generation. A new generation is formed based on the fittest individuals of the previous one. This procedure of evaluation of generation N and production of generation N+1 (based on N) is iterated until a performance criterion is met. The creation of offspring based on the fittest individuals of the previous generation is known as breeding. The breeding procedure includes three basic genetic operations: reproduction, crossover and mutation.

Reproduction selects probabilistically one of the fittest individuals of generation N and passes it to generation N+1 without applying any changes to it. Crossover selects probabilistically two of fittest individuals of generation N; then in a random way chooses a number of their characteristics and exchanges them in a way that the chosen characteristics of the first individual would be obtained by the second and vice versa. Following this procedure creates two new offspring that both belong to the new generation. Finally the mutation selects probabilistically one of the fittest individuals and changes a number of its characteristics in a random way. The offspring that comes out of this transformation is passed to the next generation. Fig. 2 illustrates the principle structure of a genetic algorithm. It starts with the random generation of an initial set of individuals, and ending with the mutation operator.

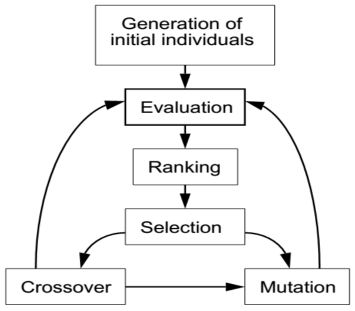

**Fig. 2. The principle structure of classical GA [10]**

## 4 Proposed GA Elman Neural Networks Learning

### 4.1 Objective function

The objective function is to choose the weight parameters of the network θ such that the squared error for data points is minimized. Thus, the objective function can be represented as follows:

$$\theta^* = \arg(\min_\theta(E(k+1,\theta)), \tag{6}$$

where,

$$E(k+1, \theta) = (z_{k+1} - y_{k+1}, \theta)^2, \tag{7}$$

$\theta$ is the vector containing the weight parameters; z(k) is the actual observed plant output. The weight parameters $W_{ch}$, $W_{ih}$ and $W_{ho}$ are updated using each collected sample according to the proposed genetic algorithm described in 4.2.

## 4.2 Proposed online genetic training algorithm

The use of online GA for training ENN to model a plant's output is illustrated in Fig. 3. A sequence of input signals $u_k$, where k  0; 1; …, is fed to both the plant and the ENN. The output signals $z_k$ for the plant and $y_k$ for the ENN are compared, and the difference E(k+ 1, θ); is computed. The error is used as a measure of the fitness of the ENN under consideration.

Each chromosome (individual) contains the weight parameters of the network θ.

$$\text{Chromosome}_{kk} = \theta_{kk} = [\text{ Wch , Wih Who}]_{kk} \tag{8}$$

where kk=1,2,3,….. population maximum number.

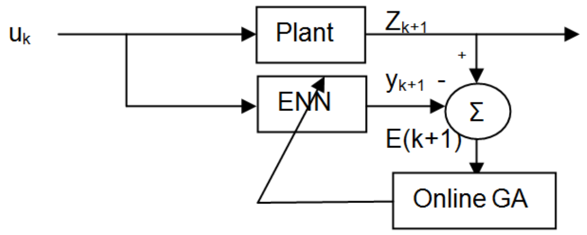

**Fig. 3. Online GA training Elman neural network**

The basic idea of this work is by inserting the best chromosomes from previous generations in the elite matrix; then we check the performance of the problem, now if the elite matrix matches the requirements then exist the GA, else perform the GA and update the elite matrix. Fig. 4 summarizes the proposed algorithm [35-38]:

### 4.2.1 General online GA

**a: Initialization:-**

1. Get the first input and output measurements.
2. Perform GA initialization by create random Population; i=1.
3. Calculate the best performance chromosome.
4. Elite Matrix[i]= best performance chromosome.

**b: For each input/ output measurements, do:**

1. Checks the Elite Matrix == required performance.
2. Test the stopping conditions: if they matching the requirement then Designate the results and get another input/output measurement. Else:
3. i=i+1;
4. Perform GA.
5. Elite Matrix[i] = best performance chromosome.

The following steps describe the details of the proposed fast genetic algorithm [35-38]:

1. Defines and initialize the variables and parameters of the ENN and GA (the most important of them are):

   NIND - number of individuals or chromosomes in the population; MAXELIT - maximum number of individuals in the elite-population; MAXGEN - the maximum number of generation for each optimization cycle; $\mathcal{E}$ - desired error value.

2. First generation initialization:

   First generation is initialized by a random real numbers. Each real number corresponds to gene in the individual (or chromosome). Number of individuals is equal to NIND, while the number of genes equals to number of variables to be optimized. The genes are randomly generated from predefined limits as follows:

   $$B_{kk}^{l} \le \theta_{kk} \le B_{kk}^{u}, \tag{9}$$

   where $kk = 1,2,....,M$ , $\theta$ is representing a gene; M is the number of genes in the chromosome; $B_{kk}^{l}$ and $B_{kk}^{u}$ are lower and upper limits of the gens respectively.

   The real value encoding scheme saves memory and improves processing speed [39-40].

3. Read the new measurements:

   The new measurements are made to clarify the current values of the objective function (fitness function) in real-time tasks.

4. Fitness evaluation of each individual in a generation:

   For each individual in the current generation, calculate the fitness function using a predefined formula or procedure.

5. Adding the best individuals in the elite population:

   Elite population, which has a maximum size equal to MAXELIT, during operation of the genetic algorithm, it is constantly formed from the individuals with the best fitness function. Further, if the elite population has MAXELIT individuals and we get an individual from the current population, which has a best fitness function value better than the current fitness function of one or more individuals from the elite population, then this individual replaces the individual having the worst fitness of the current function in the elite population.

6. Checking the termination criterion optimization of elite population:

   Decision to complete the procedure of searching an acceptable solution at the current step of the algorithm and output measurement results shall be accepted if the elite population has at least one such individual has a fitness function better than the specified precision optimization error ( $\varepsilon$ ). If such individuals are more than one, of course, select the individual which has the best fitness function. Otherwise, the decision will be on the implementation of the main stages of the classical GA.

7. Classical genetic algorithm (Fig. 4 inner loop and Fig. 3):

   The classical or conventional genetic algorithm has following major components [39]: selection of the parents - the most appropriate individuals to participate in the creation of a new generation (recombination); crossover: genes are exchanged or combined during recombination; mutation: selects probabilistically one of the fittest individuals and changes a number of its characteristics in a random way. The formation of a new generation and evaluation for its constituent is species of the fitness function. Finally, for each new generation a verifiable criterion formed at the end of the optimization process. This criterion is met in two ways: if at least one of the individuals of the current generation of fitness function is better than the specified precision optimization ( $\varepsilon$ ), i.e. convergence of GA population [40]; or turn out of maximum generation (MAGEN). With this approach, in principle, complete the loop of optimization procedure, and then quit the GA.

8. Choosing the best individuals from the current genetic population or the elite population, then output of the result:

   In this step, the current chromosome is selected from either the classical genetic algorithm population or from the elite-population individuals, which has the best fitness function. This chromosome is taken as the result of solving the problem at the current step measuring real-time algorithm.

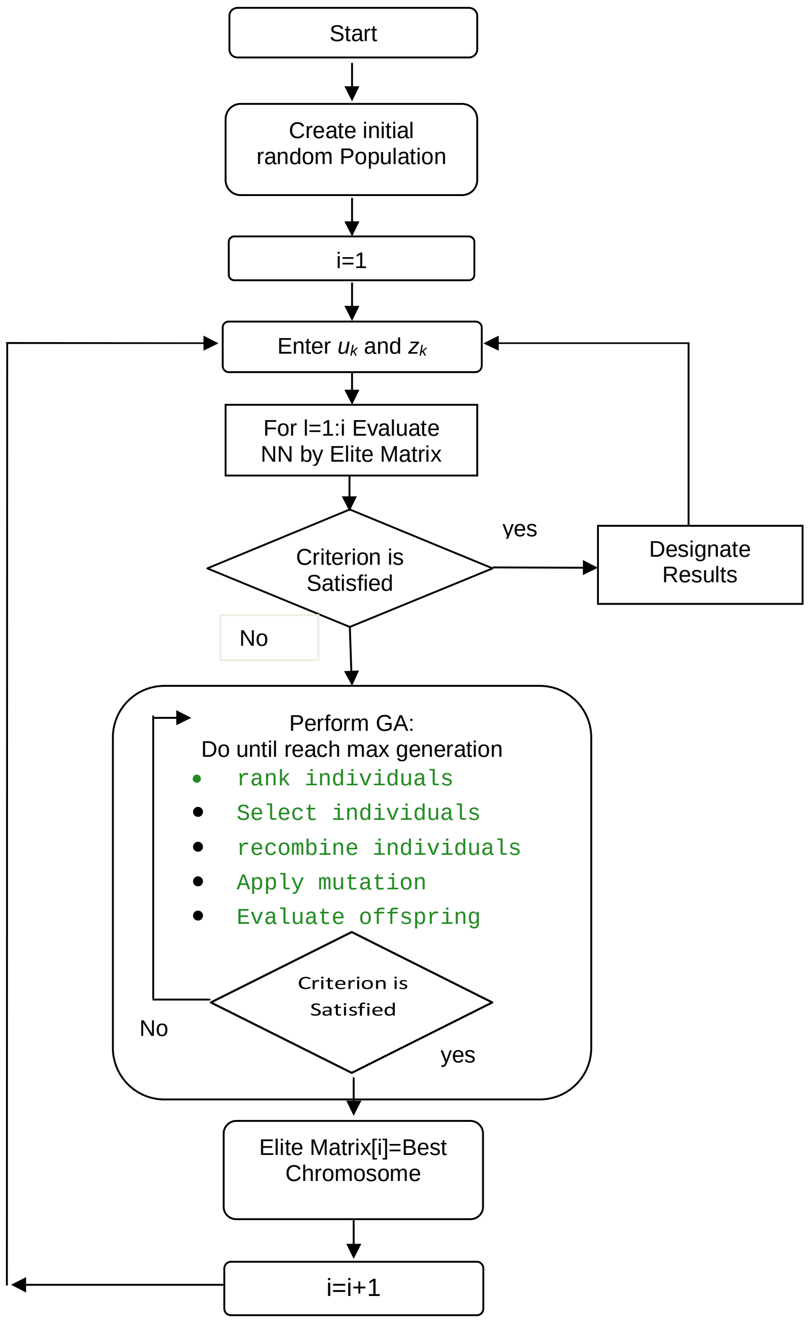

**Fig. 4. Proposed genetic algorithm**

## 5 Results

This section shows some simulation results, the training input signal for one second that is corresponding to100 sample used is of sinusoidal function:-

$$u_k=\sin(2\pi ft)\ , \qquad (10)$$

where f is the frequency of value 5 hertz.

the testing signal of one second- which is corresponding to 100 sample input samples used is:-

$$u=\begin{cases} 0 & t<0\sec \\ 1 & 0<0.25\sec \\ square(2\pi ft) & 0.25\le t<0.75\sec \\ \sin(2\pi ft) & t\ge 0.75\sec \end{cases}, \qquad (11)$$

The real number encoding is employed. The population size is set to 20 individuals, and the initial generation number is set to 1000 then reduced to 50 after one cycle.

The simulation is made on the following three plants that differ in their degree of nonlinearity [41].

Plant Number 1:- Which is a linear in both input and output behavior, as describe in equation (12).

$$z_k\text{=A1zk-1+ A2zk-2 +B1uk-1+ B2uk-2}, \qquad (12)$$

where A1=1.752821, A2= -0.818731, B1=0.011698, B2=0.010942.

The neural network is trained using extended Kalman filter for comparison purposes. Fig. 5 shows the training phase using the proposed GA; while Fig. 6 shows the plant output.

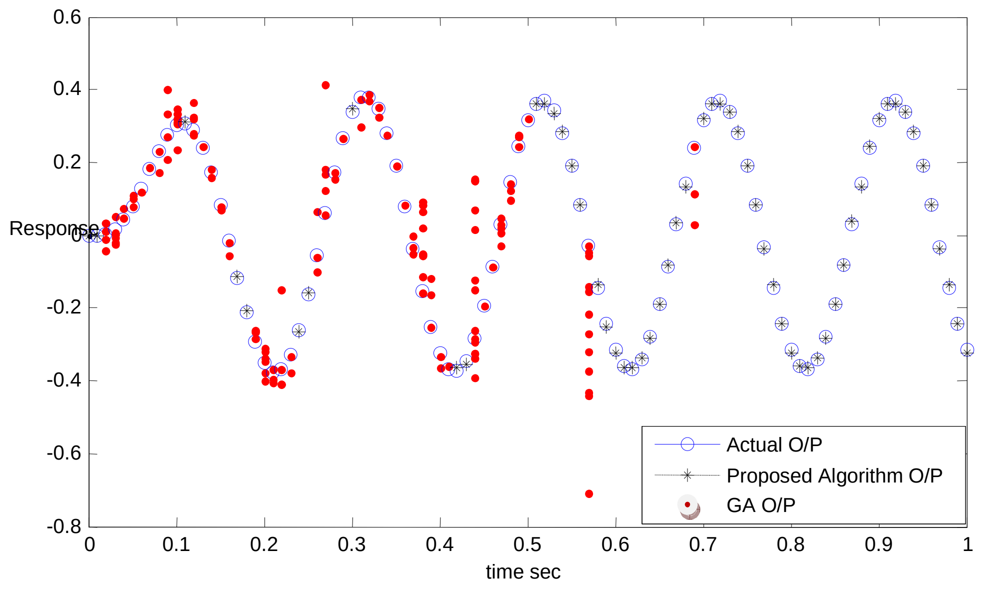

**Fig. 5. The first system training phase**

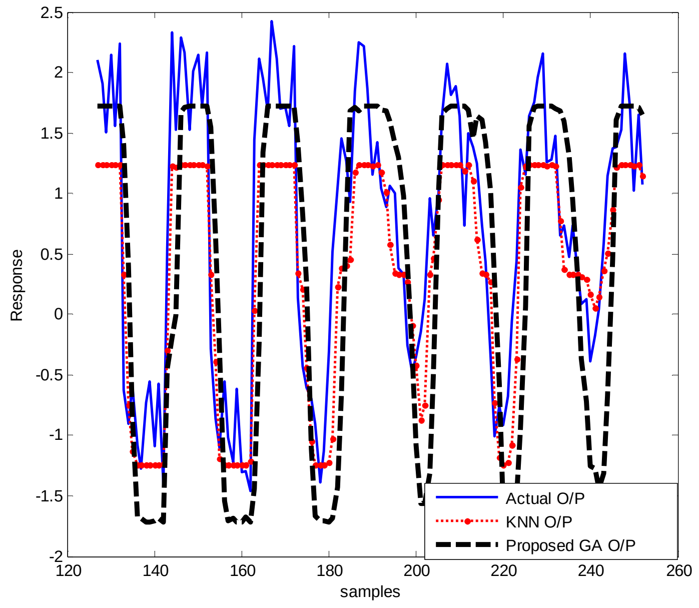

**Fig. 6. The first system output response**

Plant Number 2:- Which is a linear in the input and nonlinear in the output behaviour, as describe in equation (13).

$$z_k = A1zk_{-1} + A_2 z_{k-2} + A3z^3_{k-3} + B_1 u_{k-2}, \qquad (13)$$

where A1=1.04, A2= -0.824, A3=0.130667, B1= -0.16

Fig. 7 shows the training phase using the proposed GA; while Fig. 8 show the plant output.

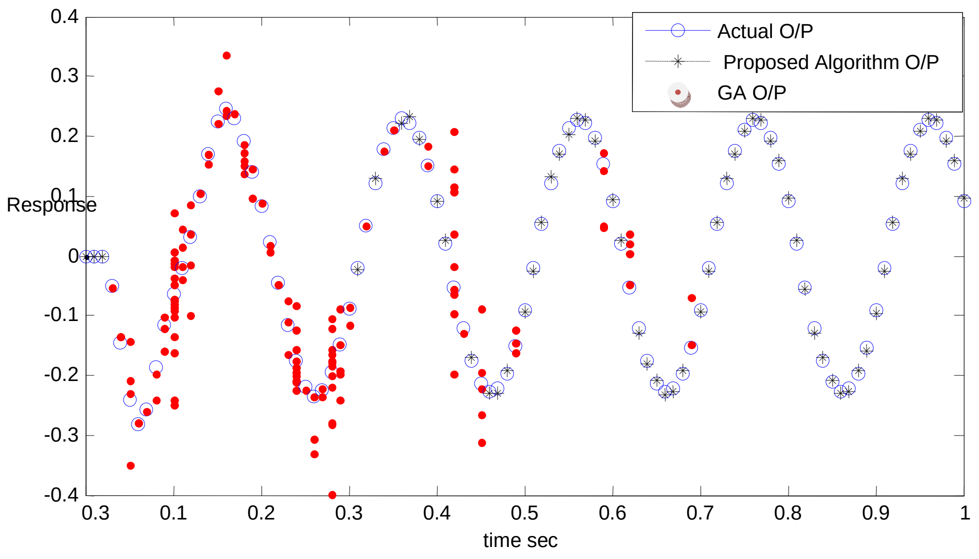

**Fig. 7. The second system training phase**

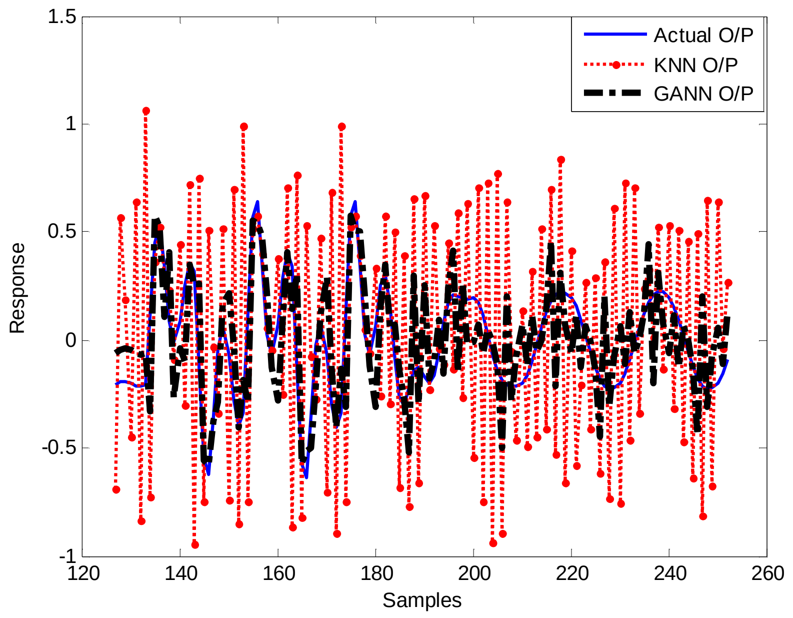

**Fig. 8. The second system output response**

Plant Number 3:- Which is a strong nonlinear in both input and in the output behaviours, as describe in equation (14).

$$z_k=A1z_{k-1}/(1+z^2_{k-1})+u^3_{k-1}, \tag{14}$$

Fig. 9 shows the training phase using the proposed GA while Fig. 10 shows the plant output.

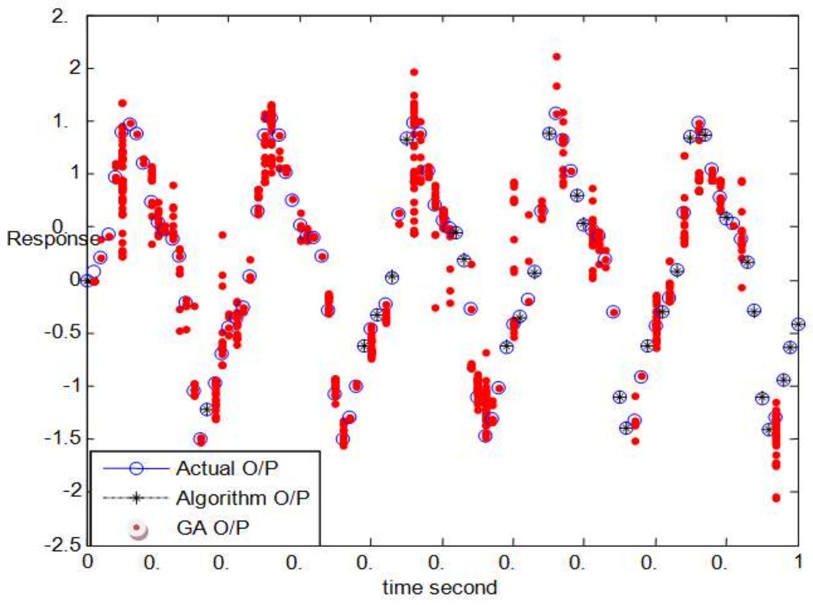

**Fig. 9. The third system training phase**

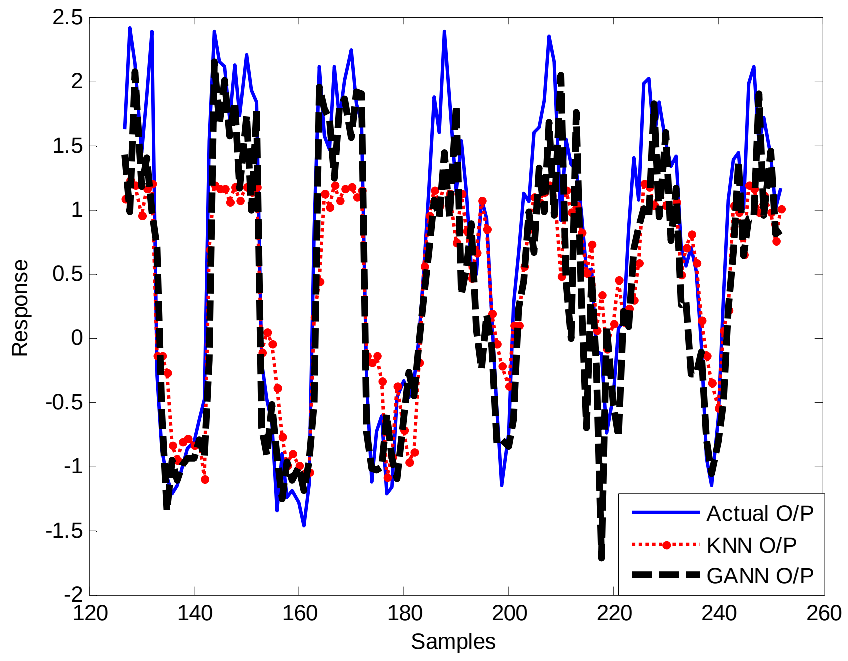

**Fig. 10. The third system output response**

## 6 Conclusion

It is well known that most NN especially Elman NN suffer from long time convergence problem as well as falling in local minima points when its optimized by either backpropagation algorithms or real time Extended Kalman filter. In this work, an online promising optimization algorithm based on GA was proposed. The algorithm allows fast convergence and prevents the search from falling into local optimum. The algorithm was simulated with three dynamical systems and different nonlinearities. The results show that the proposed approach is suitable for real time NN application. The significance of proposed algorithm is optimizing the connection weights to improve the training speed and convergence, save the running time of the neural network and improve the neural network efficiency; which can lead to improved the treatment capacity of the network in dealing with more complicated problems.

## Competing Interests

Authors have declared that no competing interests exist.

## References

[1] Gary M. Scott, Harmon Ray W. Experiences with model-based controllers based on neural network process models. Journal of Process Control. 1993;3(3):179-196.

[2] Haykin S. Neural network: A comprehensive foundation. Macmillan Publishing Company, NJ, Lehigh Press; 1994.

[3] Boden M. A guide to recurrent neural networks and backpropagation. DALLAS Project, SICS Technical Report 2002.
Available:http://www.itee.uq.edu.au/~mikael/papers/ru_dallas.pdf

[4] Yongli Wang. Optimizing of artificial neural network based on immune genetic algorithm in power load forecasting. In Proceedings of ACIIDS Posters. 2011;329-338.

[5] Miao K, Chen F, Zhao ZG. Stock price forecast based on bacterial colony RBF neural network. Journal of Qing Dao University. 2007;20:50–54.

[6] Engoziner S, Tomesn E. An accelerated learning algorithm for multiplayer perception: Optimization layer by layer. IEEE Trans. on Neural Network. 1995;6:31–42.

[7] Mitchell M. An introduction to genetic algorithms. MIT Press Cambridge, MA, USA; 1998.

[8] Holland JH. Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press; 1975.

[9] Michalewicz Z. Genetic algorithms + data structures = evolution programs. Springer-Verlag Berlin Heidelberg; 1994.

[10] Koehn P. Combining genetic algorithms and neural networks: The encoding problem. Msc. Thesis, The University of Tennessee, Knoxville; 1994.

[11] Topchy AP, Lebedko OA, Miagkikh VV. Fast learning in multilayered networks by means of hybrid evolutionary and gradient algorithms. In Proceedings of International Conference on Evolutionary Computation and its Applications. 1996;390–398.

[12] Sexton RS, Dorsey RE, Johnson JD. Toward global optimization of neural networks: A comparison of the genetic algorithm and backpropagation. Decision Support Syst. 1998;22(2):171–185.

[13] Sexton RS, Dorsey RE, Johnson JD. Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing. European Journal of Operational Research. 1999;114:589-601.

[14] Schiffmann W, Joost M, Werner R. Synthesis and performance analysis of multilayer neural network architectures. Inst. Phys., Koblenz, Univ. Koblenz, Tech. Rep. 16/1992; 1992.

[15] Topchy A, Lebedko O, Miagkikh V, Kasabov N. Adaptive training of radial basis function networks training based on cooperative evolution and evolutionary programming. In Proc. Int. Conf. Neural Information Processing (ICONIP), Dunedin, New Zealand. 1997;253–258.

[16] Gruau F. Genetic synthesis of boolean neural networks with a cell rewriting developmental process. in Proc. Int. Workshop Combinations of Genetic Algorithms and Neural Networks (COGANN-92), D. Whitley and J. D. Schaffer, Eds. Los Alamitos, CA: IEEE Computer Soc. 1992;55–74.

[17] Schaffer JD, Whitley LD, Eshelman LJ. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In Proceedings of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks, L. D. Whitley and J. D. Schaffer (eds.), IEEE Computer Society Press, Los Alamitos, California; 1992.

[18] Fogel DB. Using evolutionary programming to create neural networks that are capable of playing Tic-Tac-Toe. In International Conference on Neural Networks, IEEE Press, San Francisco, CA. 1993; 875–880.

[19] Weiss G. Towards the synthesis of neural and evolutionary learning, in: O. Omidvar (Ed.), Progress in Neural networks. 5, Ablex Pub. 1993;Chapter 5.

[20] Gupta JND, Sexton RS. Comparing backpropagation with a genetic algorithm for neural network training. Omega. 1999;27:679-84.

[21] Zhang LJ, Yuan D. Stock market forecasting research based on GA-Elman neural network. East China Econ Manag. 2008;22(9):79–82.

[22] Zhang XL, Zhu CY. Elman neural networks based on genetic algorithms and its application in prediction of MH-Ni battery capacity. Ind Instrum Autom. 2009;4:100–102.

[23] Wang T, Ye DQ. Application of improved dynamic neural network based on GA to stock market prediction. Comput Technol Dev. 2009;19(1):214–216.

[24] Montana D, Davis L. Training feedforward neural networks using genetic algorithms. In Proc. 11th Int. Joint conf. Artificial Intelligence. San Mateo, CA: Morgan Kaufmann. 1989;762–767.

[25] Jingbo Xu, Huimin Lu, Qiang Li. Optimization of process parameters of preparing foamed Al-Si alloy based on ga-based BP neural network. International Conference on Advances in Materials and Materials Processing ICAMMP 2011, Department of Metallurgical and Materials Engineering Indian Institute of Technology, Kharagpur December 9-11; 2011.

[26] Chien-Yu Huang, Chieh Huang. A robust design approach for genetic algorithm-based back propagation neural networks. Proceedings of International Conference on Business and Information (BAI2012), Sapporo, Japan; 2012.

[27] Zhang BT, Muhlenbein H. Evolving optimal neural networks using genetic algorithms with Occam’s razor. Complex Systems. 1993;7:199–220.

[28] Ioan Ileană, Corina Rotar, Arpad Incze. The optimization of feedforward neural networks structure using genetic algorithms. Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics - ICTAMI 2004, Thessaloniki, Greece.

[29] Manic M, Wilamowski B. Robust algorithm for neural network training. Proceedings of International Joint Conference on Neural Networks. IJCNN '02. 2002;2:1528 – 1533.

[30] Sexton RS, Dorsey RE, Johnson JD. Beyond backpropagation: Using simulated annealing for training neural networks. Journal of Organizational and End User Computing (JOEUC). 1999;11(3).

[31] Sexton RS, Alidaee Bahram, Dorsey RE, Johnson JD. Global optimization for artificial neural networks: A tabu search application. European Journal of Operational Research. 1998;106:570-584.

[32] Xin Yao X. Evolving artificial neural networks. Proceedings of the IEEE. 1999;87(9):1423–1447.

[33] Weiss G. Neural networks and evolutionary computation—Part 1: Hybrid approaches in artificial intelligence. In Proc. 1st IEEE Conf. Evolutionary Computa., Orlando, FL. 1994;1:268–272.

[34] Perwej Y, Perwej A. Prediction of the Bombay stock exchange (BSE) market returns using artificial neural network and genetic algorithm. Journal of Intelligent Learning Systems and Applications. 2012;4:108-119.

[35] Hasan AH, Grachev AN. Adaptive α-β-filter for target tracking using real time genetic algorithm. J. Electr. Control Eng. (JECE). 2013;3:32–38.

[36] Hasan AH, Grachev AN. On-line parameters estimation using fast genetic algorithm. Journal of Electrical and Control Engineering (JECE). 2014;4(2):16-21.
Available:www.joece.org/ © American V-King Scientific Publishing

[37] Hasan AH, Grachev AN, Lukashenkov AV, Fomichev AA. Simultaneous estimation of states and parameters using Kalman filter and fast genetic algorithm. Journal of Control Engineering and Technology (JCET). 2014;4(2):153-159.

[38] Hasan AH, Grachev AN. Target tracking by adaptive EKF using fast genetic algorithm. International Journal of Information Engineering (IJIE). 2014;4(2):46-52.

[39] De Jong KA, Spears WM, Gordon DF. Using genetic algorithms for concept learning. Machine Learning. 1993;13:161-188.

[40] Janikow CZ. A knowledge-intensive genetic algorithm for supervised learning. Machine Learning. 1993;13:189-228.

[41] Pham DT, Xing L. Neural networks for identification, prediction and control. Springer-Verlag, 2nd; 1995.

***Peer-review history:***
*The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)*
*http://sciencedomain.org/review-history/16534*