Nuclear Research Reactors Accidents Diagnosis Using Genetic Algorithm/Artificial Neural Networks

Abdelfattah A. Ahmed, Nwal A. Alfishawy*, Mohamed A. Albrdini* and Imbaby I. Mahmoud****

* Dept of Comp. Sci. & Eng., Faculty of Electronic Engineering, Minufiya University, Minuf, Egypt.
** Atomic Energy Authority, Nuclear Research Center, Inshas, Egypt.
fatt231153@yahoo.com

Abstract: The Nuclear Research Reactors plants are expected to be operated with high levels of reliability, availability and safety. In order to achieve and maintain system stability and assure satisfactory and safe operation, there is increasing demand for automated systems to detect and diagnose such failures. In recent years, both Genetic algorithms and neural networks, which are inspired by computation in biological systems, are emerged as established techniques for optimization and learning. Genetic algorithms have been used in conjunction with neural networks in three major ways: First, genetic algorithms have been used to construct neural network topologies. Second, they have been used to set the weights in fixed architectures. Third, they have been used to select training data and to interpret the output behavior of neural networks. This paper is concerned with the construction of Artificial Neural Networks (ANNs) using Genetic algorithms (GAs) for the nuclear accidents diagnosis. MATLAB ANNs toolbox and GAs toolbox are employed to optimize an ANN for this purpose. When we apply the results obtained from genetic algorithms on the back-propagation algorithm, the results are similar but the design of ANNs using GAs is useful in terms of automating and optimizing the design and finding weights and biases for the suggested construction. The results obtained show the efficiency of using genetic algorithm, which can construct the high performance neural network structure for the nuclear reactor's input data. The best structure obtained is two layers ANN with correspondence values of weights and biases that are required to construct such network. [Abdelfattah A. Ahmed; Nwal Ahmed Alfishawy; Mohamed A. Albrdini, and Imbaby I. Mahmoud. Nuclear Research Reactors Accidents Diagnosis Using Genetic Algorithm/Artificial Neural Networks. Nature and Science 2011; 9(5):64-74]. (ISSN: 1545-0740). http://www.sciencepub.net.

Keywords: Genetic algorithms (GA), Artificial Neural Networks (ANN), Nuclear Reactors, Accidents Diagnosis, MATLAB.

1. INTRODUCTION

As the systems become more complex and they need to operate with minimum malfunctioning or breakdown time, reliable automated fault detection and diagnosis system are increasing rapidly. The early detection of plant's failure could prevent system malfunction or serious damage, which could also lead to disaster. Therefore, an intelligent fault detection and diagnosis system to deal with inaccurate information has also been greatly required [1-5].

Genetic Algorithms and Neural networks are two techniques for optimization and learning, each with its own strengths and weaknesses. The two have generally evolved along separate paths. However, recently there have been attempts to combine the two technologies. The genetic algorithms implemented in the toolbox let to solve optimization problems with nonlinear, linear, and bound constraints. The genetic algorithm improves the chances of finding a global solution, due to its random nature. [5-10].

In this paper, diagnoses of nuclear reactor accidents are investigated. A computer program is developed using MATLAB environment for this purpose. Genetic algorithm routine is implemented and employed to construct an artificial neural network, to diagnose the nuclear reactor accidents. The program is capable of plotting the various relations between the neural network output and the required target. The program stops calculation when the mean square error or sum square error reaches a desired minimum or met one of the default stopping criteria. The proposed NN constructed by GA outperforms backpropagation alone, which is the standard training algorithm, on our reactor accidents data case. This performance gain comes from tailoring the genetic algorithm to the domain of training neural networks (reactor accidents data). The results of this program are demonstrated by a series of screens shots and output plots. Comparing with results published in [1], a significant improvement is obtained. This paper is organized as follows: Section 2 presents the problem formulation. Section 3 describes the proposed system. Section 4 shows the results and discussions of the proposed system performance. Section 5 is devoted to conclusion.

2. PROBLEM FORMULATION

The system is designed to construct an ANN using Genetic algorithms (GAs) for accidents diagnosis of the nuclear reactor's input data. MATLAB toolboxes

are used to implement the GA which produces the optimized values of weights and biases that are required to construct such network. The data used in the application were collected by the aid of reactor operation crew and Safety Analysis Report (SAR) of the reactor, in addition to the Atomic Energy experts. The data sets are for the eight accidental cases (Classes) listed below; plus the normal operation case as shown in Figure (1). So the total cases, which we have, are nine. The result is that each accident is represented as a 15 by 1 grid of Boolean values [1, 2].

```
[0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]
[1 1 0 1 1 0 0 1 1 1 1 1 1 1 1]
[1 1 1 1 1 1 1 0 1 1 0 1 1 1 1]
[1 1 1 1 1 1 1 0 0 1 0 0 1 1 1]
[0 0 0 0 1 1 1 0 0 0 1 0 1 1 1]
[1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]
[1 1 0 0 1 1 1 1 1 1 1 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

Figure (1): Sample of reactor accidents data patterns.

In optimization problems a set of parameters is selected that will give the best solution to a particular problem. To present the initial values of these parameters to a GA, it must be encoded into a string so that crossover and mutation can be applied. Binary encodings are the most common, due to the fact that Holland [9] used them in his early pioneering work. In choosing an encoding scheme the nature of the problem will play a major role. So, in the reactor accidents diagnosis, the binary encoding becomes practical as the accidents patterns are in binary form. Any base can be used, as it is just a different method of encoding the same information, but the lower the base the longer the string will be [6-9].

3. PROPOSED SYSTEM

In this section, we will explain the proposed system modules. It is consisted of an m file program for GA computation and a Graphical User Interface (GUI) for easy communication with the program and experimentation with alternatives. Matlab environment is used to implement the system.

3.1 Program Description

The graphical user interface (GUI) is depicted in figure (2), which is used to guide the user for the optimization process.

Calculation is started by clicking *'Calculate'* button, where the calculation program begins with calculating the different parameters (the weights and biases, the transfer functions and the performance functions) for the two and the three layers of the ANN.

3.2 The Implemented GA.

A block diagram of the implemented GA is shown in Figure (3). The genetic algorithm begins by creating a random initial population. If the minimal point for the fitness function is known approximately where it lies, the initial range should set so that the point lies near the middle of that range. However, the genetic algorithm can find the minimum even with a less than optimal choice for initial range.

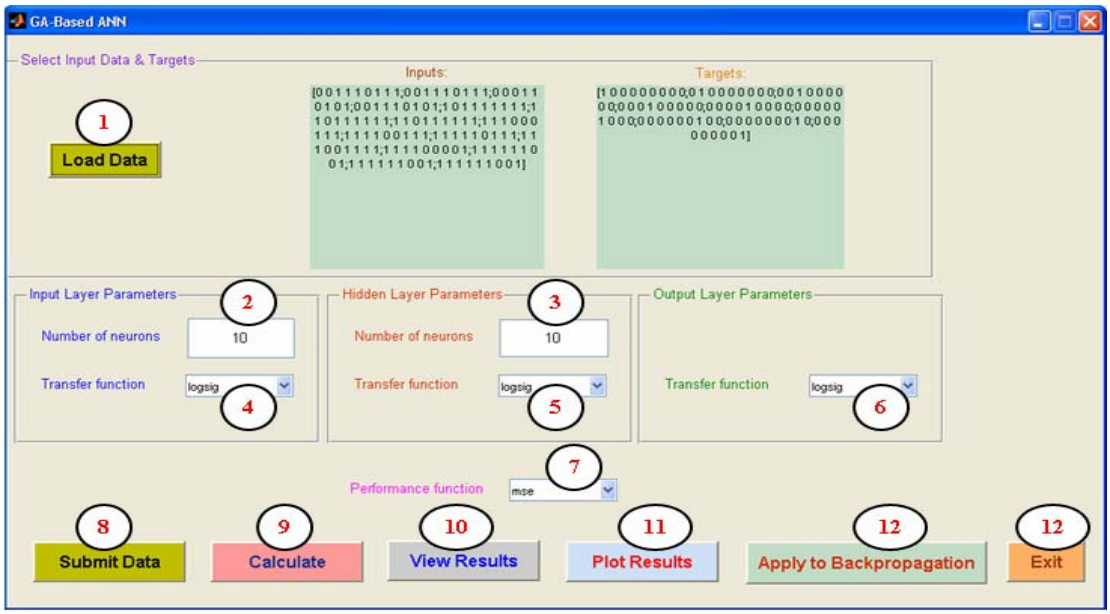

Figure (2): The program graphical use interface (GUI)

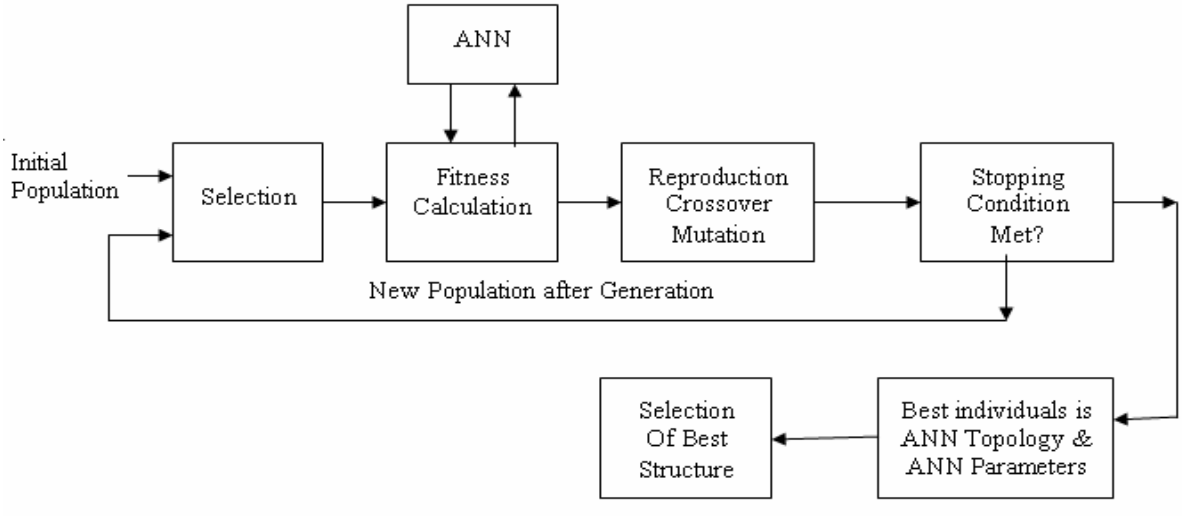

Figure (3): Artificial Neural Network and Genetic Algorithm

A genetic algorithm is used to find a good topology and parameters for a neural network. Using neural network as the *fitness function*, GA determines the fitness level of each *topology and parameters*. Using fitness values, the genetic algorithm would then evolve a new population for the network to try. After several generations, a population of several "good" structures with parameters evolves and fittest topology and parameters are used as the best construction of the neural network.

The genetic algorithm uses the individuals in the current generation to create the offspring that make up the next generation. Then, the algorithm creates a sequence of new populations. To create the new population, the algorithm scores each member of the current population by computing its fitness value. The Fitness Value of the individual Vi of GA is calculated as follows:

- The chromosome V_i string and the corresponding settings of ANN with reference the choices from the program GUI.
- Designed ANN model as per the string and the network choices. Predicted output y_j is given as $y_j = f(x_j, W, b)$, where x_j is input vector, W is weight matrix and b is bias matrix.
- Performance is checked by calculating fitness values and is expressed in terms of Mean-Squared Error (*MSE*) or Sum-Squared Error (according to the choice of the performance function from GUI) as:

$$SSE = \sum_{i=1}^{N_p} \sum_{k=1}^{K} (t_{i,k} - y_{i,k})^2 \quad \text{--(1)}$$

$$MSE = \frac{1}{N_p K} \sum_{i=1}^{N_p} \sum_{k=1}^{K} (t_{i,k} - y_{i,k})^2 \quad \text{--(2)}$$

Where N_p and K denote the number of patterns and output nodes used in the training respectively, i denotes the index of the input pattern (vector), k denotes the index of the output node, $t_{i,k}$ and $y_{i,k}$ express the desired output (target) and actual output values of the k^{th} output node at i^{th} input pattern, respectively. The calculation of the output is according to figure (4) for two layers network using equation (3) and figure (5) for three layers network using equation (4)

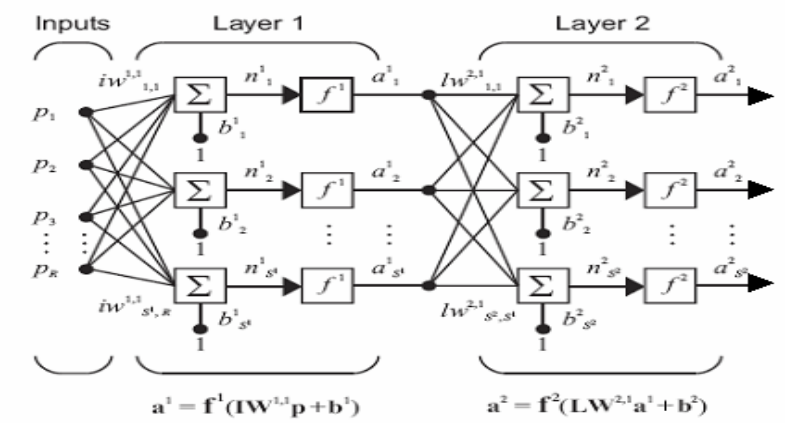

Figure (4): The output of two layers Artificial Neural Network structure

$$a^2 = f^2\,(LW^{2,1}\,(f^1\,(LW^{1,1}p + b^1) + b^2) = y_j\,, \qquad (3)$$

where j is neurons in the output layer,

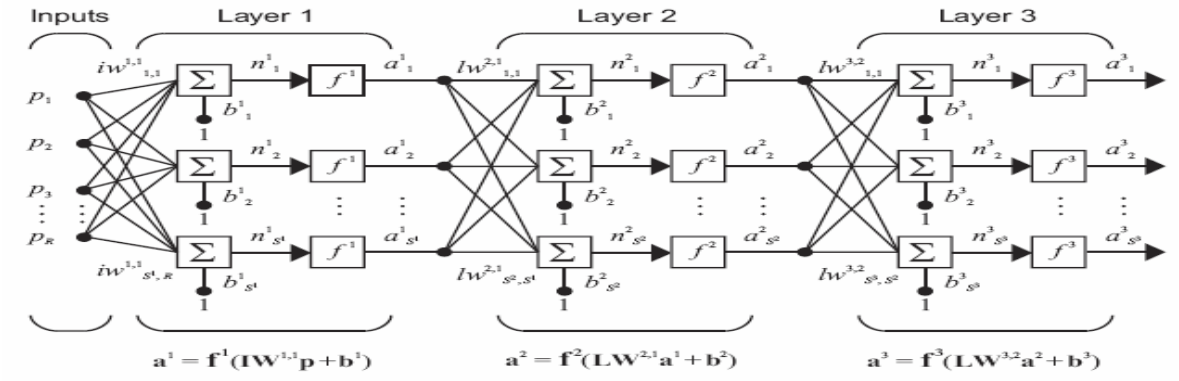

Figure (5): The output of three layers Artificial Neural Network structure

$$a^3 = f^3\,(LW^{32}\,f^2(\,LW^{2,1}\,(\,f^1\,(\,LW^{1,1}p + b^1\,) + b^2) + b3) = y_j\,, \text{--(4)}$$

where j is neurons in the output layer,

The setting for GA is shown in Table (1). Based on speed consideration some values set for population size and evaluation generation a little small.

3.2.1 Algorithmic Setting

Table (1): Parameters setting for GA.

Setting Type	Value
Encoding Scheme	Binary Encoding
Population Size	100
Selection	Stochastic Uniform
Crossover	Uniform
Mutation	Random
Elitism	Yes
Stall Time Limit	20
Display	iteration

a) The **selection** function chooses parents for the next generation based on their scaled values from the fitness scaling function. An individual can be selected more than once as a parent. The default selection option, *stochastic uniform*, lays out a line in which each parent corresponds to a

section of the line of length proportional to its scaled value. The algorithm moves along the line in steps of equal size. At each step, the algorithm allocates a parent from the section it lands on.

b) Some of the individuals in the current population with the best fitness values are chosen as *elite*. The default setting is 10% of the current population size. These *elite* individuals are passed to the next population. Setting Elite count to a high value causes the fittest individuals to dominate the population, which can make the search less effective.

c) Besides elite children, the algorithm:

- Creates *crossover* children by selecting vector entries, or genes, from a pair of individuals in the current generation and combines them to form a child. The default setting is 80% of the current population size, after excluding the elite children. With some probability *P* (typically between 0.5 and 0.8), it is decided how parent contribute to the gene values in the offspring chromosome. For example:
  Parent1 : 11001010
  Parent2 : 00110111
  If *P* is 0.5 :
  Offspring 1 : Parent1 contributes ***odd*** positions, Parent2 contributes ***even*** positions
  Offspring 2 : Parent1 contributes ***even*** positions, Parent2 contributes ***odd*** positions
  This is means approximately half of the genes in the next offspring will come from Parent1 and the other half will come from Parent2. Example:
  - Contribution of Parent1 : 1 0 1 1
  - Contribution of Parent2 : 0 0 1 1
  ----
  * New Offspring1 : 10001111
  * By the same way New Offspring2 : 01100010
- Creates *Mutation* children by applying random changes to a single individual by altering one or more genes in the chromosome from its initial state. The default setting is 20% of the current population size, after excluding the elite children. The mutation process is illustrated as the following:
  7 6 5 **4** 3 2 1 0 bit position
  Before Mutation : 1 0 1 **1** 0 1 1 1
  After Mutation : 1 0 1 **0** 0 1 1 1
  Random mutation exchanges a random selected gene with a random value within the range of the genes minimum and maximum value.
- Crossover rate generally should be high, and mutation rate should be very low

d) Replaces the current population with the children to form the next generation.

e) The stopping criteria, as Generations, Time limit, Fitness limit, Stall generations (The algorithm stops when the weighted average change in the fitness function value over Stall generations is less than Function tolerance), Stall time limit (The algorithm stops if there is no improvement in the objective function during an interval of time in seconds equal to Stall time limit.), Function Tolerance (The algorithm runs until the weighted average change in the fitness function value over Stall generations is less than Function tolerance.) and Nonlinear constraint tolerance (The Nonlinear constraint tolerance is not used as stopping criterion. It is used to determine the feasibility with respect to nonlinear constraints.), stops the algorithm as soon as any one of these conditions is met. You can specify the values of these criteria in the Stopping criteria pane in the Optimization Tool or by the function 'gaoptimset' from the command line.

4. RESULTS AND DISCUSSION

4.1. Results of Constructed ANN by GA

The output response, after finishing the calculation, show a 9-by-9 matrix with diagonal values larger than 0.9, as in the figure (6-a). Then, when these diagonal values rounded, it approximately equal 1, as in the figure (6-b), and that is exactly the required target. This means that the constructed neural network is the best for the reactor accidents data. While the best structure is obtained for the ANN, the correspondence values of weights and biases, that are required to construct such network, are also calculated by the program. Figures from (7-1) through (7-4) display the calculated output provided by the proposed system. Figure (7-1) show layer1 weight matrix 10x15, that is required to construct layer one. Figure (7-2) show layer1 bias matrix 10x9, that is required to construct layer one. Figure (7-3) show layer2 weight matrix 9x10 that is required to construct layer two. Figure (7-4) show layer2 bias matrix 9x9, which is required to construct layer two.

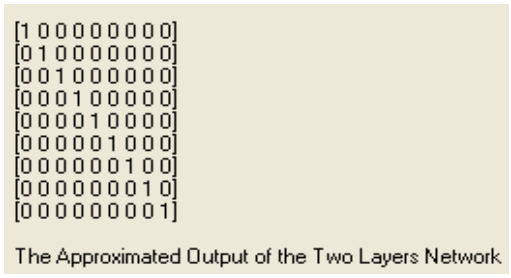

Figure (6-a):

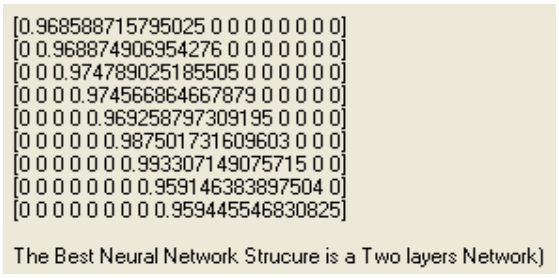

Figure (6-b):

1.294174	0.403967	0.022914	0.244113	0.616213	0	0	0.55837	0.852117	0.220866	0.100933	0	1.338927	0.308076	0.159786
0.718047	0.114096	0.052566	0.104471	0.093289	0.10646	0.10646	0.596243	0.143794	0.168883	0.095424	0.116833	0.101025	0.015186	0.190239
0.119424	0.121144	1.467614	0.196043	1.05597	0.666791	0.107896	0.546831	0.134914	0.109544	0.701517	0.139553	0.339345	0.095098	0.921141
0.002896	0.237737	0.134907	0.821212	0.007631	0.354783	0.006918	0.004494	0.030706	0.004918	0.808678	0.607187	0.003464	0.003464	0.634602
0.232903	0.24205	0.24126	0.771206	0.396973	0.410102	0.410102	0.877658	0.303567	0.405242	0.312016	0.506578	0.460419	0.460419	0.490105
0.896784	1.11265	1.020514	6.09E-06	0.411697	8.71E-05	1.1482	0.342579	1.000721	0.020681	0.175393	0.061857	0.228265	4.93E-05	0.245899
0	0	0.105134	0	0	0	0.120325	0.13839	0.445451	0.43366	0.809465	0.809116	0	0.81693	0.813953
0.41986	0	0	1.18536	0.001899	2.22027	0	0.562003	0.133793	0.496002	0.15001	0.004008	0.505742	0	0
0	0	0	0	0	0.000324	0.072711	0	0	0	0.081883	0	0	0	0.203292
0.842985	0.001318	0.287045	0.741289	0.547496	0	1.04216	0.696164	0.749939	0.657586	0.002687	0	0.001152	0.738568	0.428855

Figure (7-1): Layer1 weight matrix (IW11) 10x15

4.69E-06	0.821539	0.22262	0.000174	0	0.113669	0.341775	0	0
0.016709	0.186577	0.023529	0.019666	0.297388	1.23353	0.02192	1.204812	0
1.136521	0.019397	0.043312	0.322597	0.593459	0.043481	0.258947	0.615361	0.520055
0.000974	0.07822	1.316873	0.488916	1.6907	0.34788	0.534693	0.003247	0.978904
0.855429	0.785634	0.315264	0.12171	1.303084	0.297257	1.475311	2.56E-06	0.23909
0.688354	1.090398	0.8214	0.67968	0.516206	8.85E-05	1.302364	0.391155	1.968722
0.461741	2.115235	0.826562	0.964289	0	0.668071	0	0.765233	0.84226
0.034372	0.366686	0.27244	0.764833	0.344254	0.336583	0.74381	0.527595	1.80E-06
0.903312	0.000124	0.367796	0.05299	0.785721	0.030262	0.018436	0.849421	0.81607
0.766573	0.000381	7.37E-09	0.536192	0.892741	0.804116	0.000496	2.02E-06	0.198955

Figure (7-2): Layer1 bias matrix (bi1) 10x9

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0.819534	0.261904	0	0.662799	0.539179	0	1.520552	2.64707	0.779306	0.738034
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure (7-3): Layer2 weight matrix (LW11) 9x10

2.843221	0	0	0	0	0	0	0	0
0	5	0	0	0	0	0	0	0
0	0	2.922706	0	0	0	0	0	0
0	0	0.349029	6.40E-05	0	0	0.191728	0.441004	0.908459
0	0	0	0	3.37511	0	0	0	0
0	0	0	0	0	4.089659	0	0	0
0	0	0	0	0	0	2.945033	0	0
0	0	0	0	0	0	0	2.871964	0
0	0	0	0	0	0	0	0	2.877407

Figure (7-4): Layer2 bias matrix (bi2) 9x9

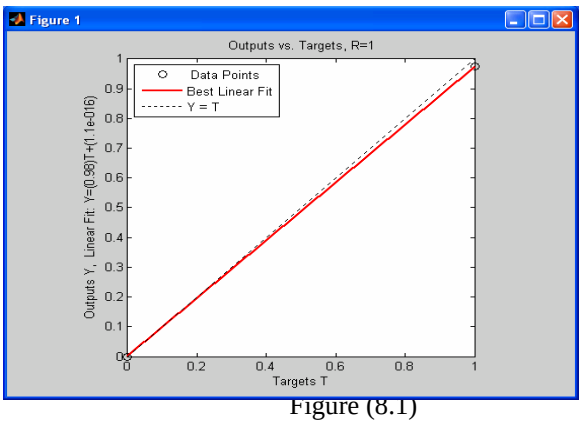

Figure (8.1)

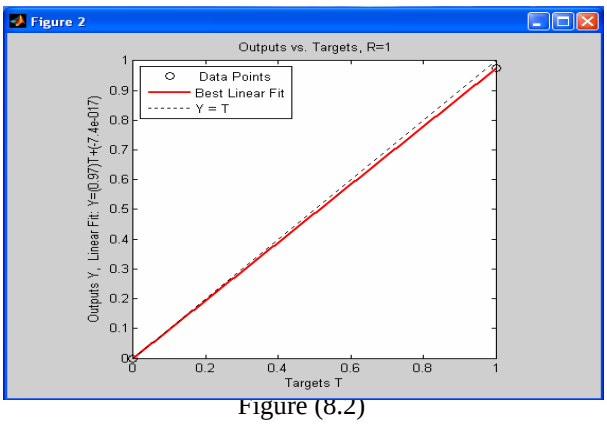

Figure (8.2)

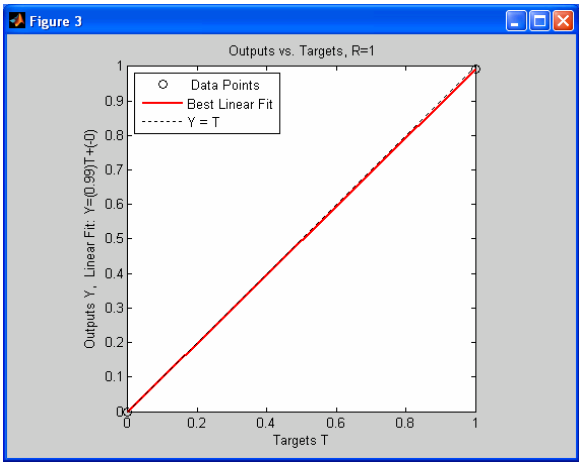

Figure (8.3)

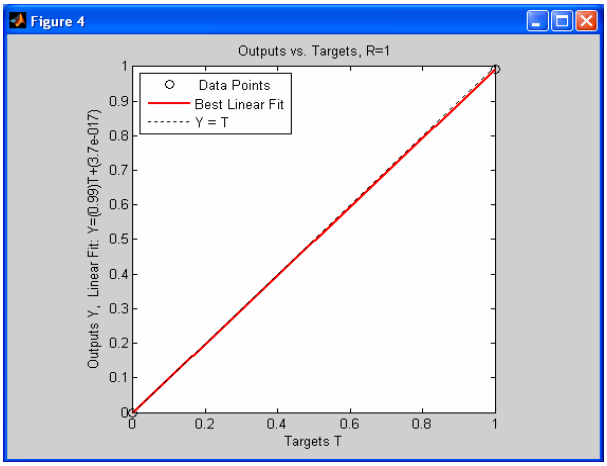

Figure (8.4)

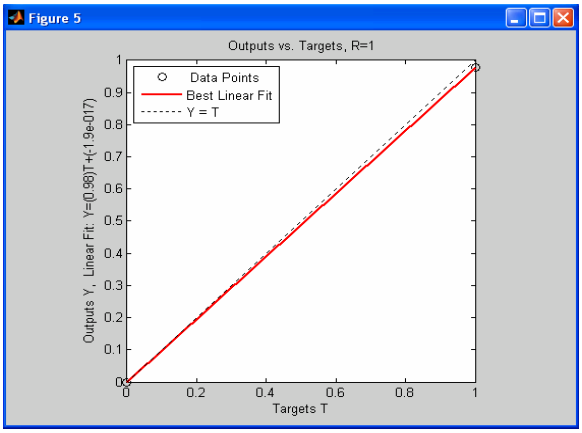

Figure (8.5)

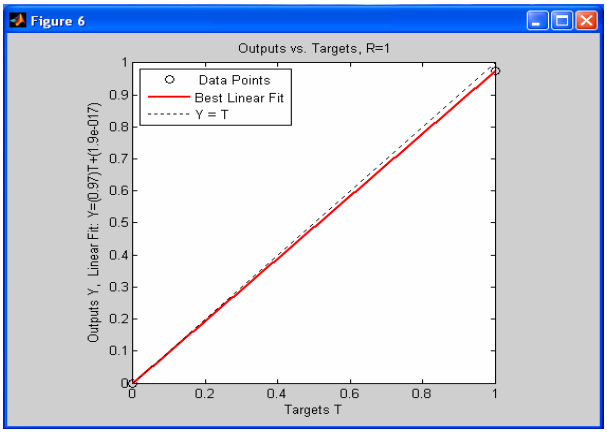

Figure (8.6)

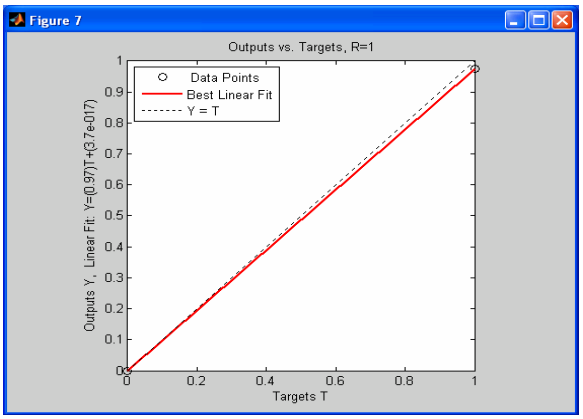

Figure (8.7)

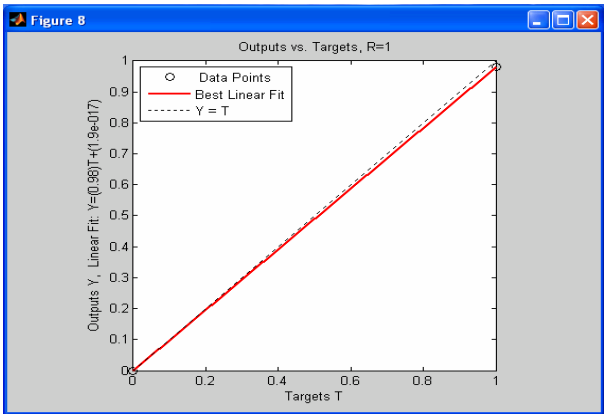

Figure (8.8)

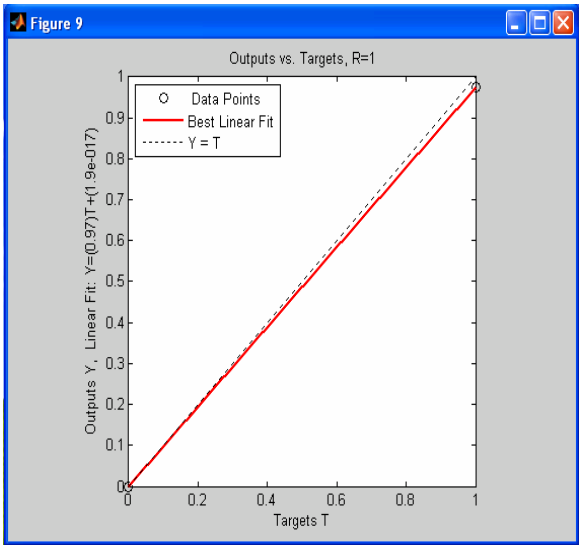

Figure (8.9)

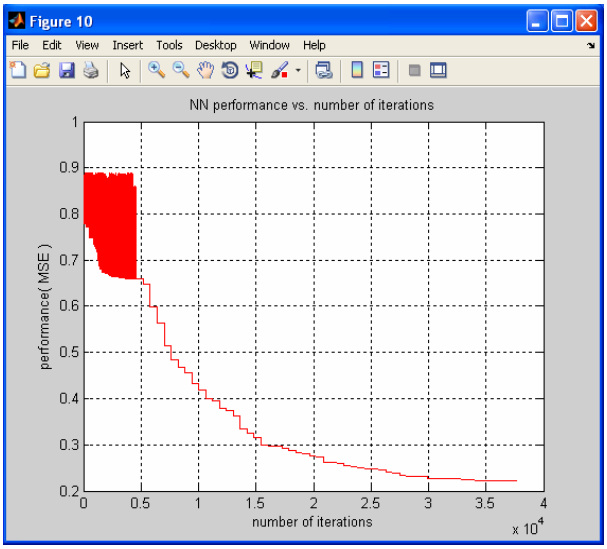

Figure (9): ANN performance vs. Number of iterations

The performance of a constructed network can be measured to some extent by investigating the network response in more detail. A regression analysis between the network response and the corresponding targets is proposed.

The network output and the corresponding targets are passed to regression analysis function, and it returns three parameters. The *first one* is the slope (m), the second is the y-intercept (b) of the best linear regression relating targets to network outputs and the *third* variable is the correlation coefficient (R-value) between the outputs and targets. If there were a perfect fit (outputs exactly equal to targets), the slope would be 1, and the y-intercept would be 0. In our program, we can see that the numbers are very close. It is a measure of how well the variation in the output is explained by the targets. If this number is equal to 1, then there is perfect correlation between targets and outputs. In clear from figure (4-a) and figure (4-b), the number is very close to 1, which indicates a good fit.

The following figures from (8.1) through (8.9) illustrate the graphical output provided by regression analysis function. The constructed neural network outputs are plotted versus the targets as open circles. The best linear fit is indicated by a dashed line. The perfect fit (output equal to targets) is indicated by the solid line. In these figures, it is difficult to distinguish the best linear fit line from the perfect fit line because the fit is so good. Is also clear that the outputs seem to track the targets reasonably well, and the R-values are equal 1.0, this means that there is perfect correlation between targets and outputs. Accordingly, this is another demonstration for the suitability of the suggested construction of neural network using genetic algorithm.

Figure (9) displays another performance measure to the constructed neural network using genetic algorithm. The result shown in this figure is reasonable, where in the first iterations from 0 to $0.5x10^4$, there is a big variation in performance (MSE) error and the error is still below the value 0.9, and it doesn't appear that any significant variation has occurred after iterations point $0.5x10^4$.

4.2 Comparing Results to Backpropagation Algorithm ANN

In this section, the parameters obtained from the construction of the neural network optimized by genetic algorithm; such as number of layers, number of neurons in each layer, activation function in each layer, performance function and the training function are compared to traditional backpropagation algorithm. Also, the initial conditions

were taken into account such as the mean of sum-squared error goal, momentum constant and number of epochs, to investigate the effect of these parameters on the network performance. Figure (10) and figure (11) show the plot of performance versus number of epochs.

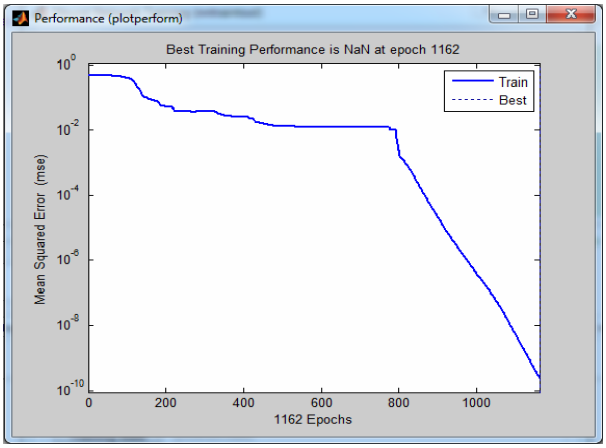

Figure (10): performance vs. number of epochs (GA parameters applied to backpropagation

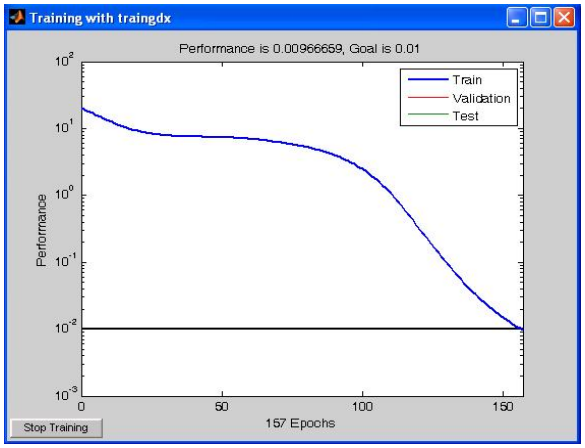

Figure (11): performance vs. number of epochs (Running backpropagation alone)

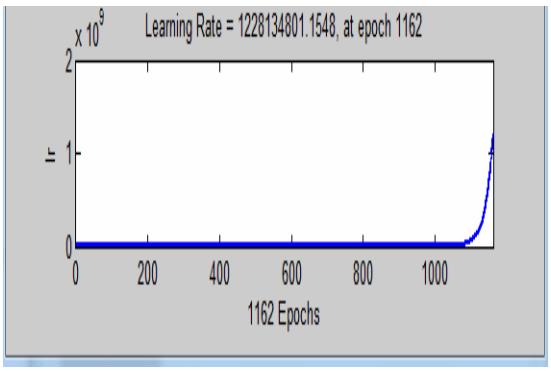

Figure (12): learning rate vs. number of epochs (GA parameters applied to backpropagation)

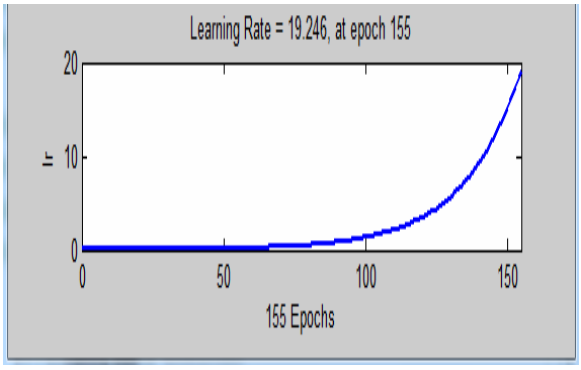

Figure (13): learning rate vs. number of epochs (Running backpropagation)

Table (2): Comparison between results when applying GA parameters to backpropagation and running backpropagation alone

Statement	**GA parameters applied to backpropagation results issued at epoch *1162***	**running backpropagation alone results issued at epoch *155***
Performance	$2.42\ e^{-10}$	$959.00e^{-5}$
Gradient	$9.90\ e^{-11}$	$5.37\ e^{-5}$
Learning rate	1228134801.1548	19.246

From the result shown in figure (9), the best performance (MSE) is 0.22 and has occurred after iterations point $3.5x10^4$. This means long time is needed for the GA to produce results, but it is acceptable with the current advances in computer technologies. From figure (12), figure (13) and Table (2), the difference between the values of the mean square error (performance), that results from applying GA parameters to backpropagation and running backpropagation alone, (using Matlab[11]), is highly large (approximately double of value). Also, for the gradient, the difference is highly large (approximately double of value), but this comes in the expense of large learning rate. This means much faster convergence when using backpropagation algorithm, ($959.00e^{-5}$ at epoch 155), but it is the best when using GA, ($2.42\ e^{-10}$ at epoch 1162).

The valuable benefits from GA are automating and optimizing the design; and finding weights and biases for a suggested construction of Artificial Neural Networks as explained in section (4.1).

5. CONCLUSION

In this paper a method was proposed for constructing an optimized neural network by employing genetic algorithm. In this method we implemented a genetic algorithm which can construct the high performance neural network structure for a given input nuclear reactors measured data, and the corresponding target accident. This method can be applied to any problem, where a data of inputs and outputs has the same form. Moreover, in this paper we present encoded nuclear reactors data (into a string), so, the genetic algorithm can be applied. In the reactor accidents diagnosis, the binary encoding becomes practical as the accidents patterns are in binary form. A comparison is conducted between a GA constructed ANN, GA initialized ANN and traditional backpropagation ANN. By applying the results obtained from genetic algorithms as initialization values on the backpropagation algorithm, the results showed much faster training, better convergence (smaller mean square error). This work demonstrated the method of automating and optimizing the design and finding weights and biases for a suggested construction of Artificial Neural Networks by Generic Algorithm in the domain of nuclear reactors.

REFERENCES:

1. S. Sh. Haggag, PhD Thesis, ***"Design and FPGA-Implementation of Multilayer Neural Networks With On-chip Learning"***, Atomic Energy Authority, Egypt 2nd Research Reactor. PhD, Menufia University, 2008.
2. The International Atomic Energy Agency (IAEA) Safety Series, "Safety in the Utilization and Modification of Research Reactors", IAEA publications, STI/PUB/961, Vienna, Austria, December 1994, STI/PUB/961.
3. J. Korbicz, Z. Kowalczuk, J. M. Koscielny, W. Cholewa : "Fault diagnosis: models, artificial intelligence, applications", ISBN 3-540-40767-7, Springer-Verlag Berlin Heidelberg 2004.
4. B. Ch. Hwang, "Fault Detection and Diagnosis of a Nuclear Power Plant Using Artificial Neural Networks", Simon Fraser University, March 1993.
5. L. S. Admuthe and S. D. Apte, "Neuro – Genetic Cost Optimization Model: Application of Textile Spinning Process", International Journal of Computer Theory and Engineering, 1793-8201, Vol. 1, No. 4, October2009.
6. I. Kusco and Ch. Thornton, "Design of Artificial Neural Networks Using Genetic Algorithms: review and prospect", Cognilive and Cornpuling Sciences, University of Sussex, Brighton BNl 9QN, April 30, 1994.
7. R. Hamdi, M. Bedda, "Arabic Speech Synthesis Using Optimized Neural Networks with Genetic Algorithms", Department of Electronic, Faculty of the Engineer, Annaba University, Algeria, Asian Journal of information Technology 5(7): 686-690, © Medwell Online, 2006.

8. A. Fiszelew, P. Britos, A. Ochoa, H. Merlino, E. Fernández, R. García-Martínez, "Finding Optimal Neural Network Architecture Using Genetic Algorithms", Software & Knowledge Engineering Center. University of Buenos Aires.Kb, Research in Computing Science 27, 2007.
9. A. M. Sadeq, A. A. Wahdan, H. M. K. Mahdi, "Genetic Algorithms and its Use with Backpropagation Network", Faculty of Engineering, Ain Shams University; Vol. 35, No. 3, Sept 30, 2000.
10. WHITLEY, "Genetic Algorithms and Neural Networks", Editor J. Periaux and G. Winter, ©1995 John Wiley & Sons Ltd. (jwbook.sty v3.0, 12-01-1995)
11. The Language Technical Computation (MATLAB), http://www.mathworks.com/, The MathWorks Inc.

3/12/2011