Robust Algorithm for Neural Network Training

Milos Manic, Bogdan Wilamowski

University of Idaho Boise Center, College of Engineering,
800 Park Blvd., Suite#200, Boise, ID 83712
{misko, wilam}@ieee.org

Abstract – Neural networks have been proven to be very successful in many cases where other traditional techniques failed to give satisfactory results. Despite their popularity, several problems exist. Even with the adequate network architecture, frustrating problems of correct choice of initial weights for given architecture remain. The proposed method uses combination of approaches used in genetic algorithms and gradient methods. Genetic algorithm is used in search for an adequate weight set for a complex error surface. Once it is done, the algorithm automatically shifts to gradient type of method. The proposed algorithm does not explicitly calculate gradients like in error back propagation. It rather estimates the gradient from the set of random feed forward calculations. The proposed approach automatically searches for the adequate initial weight set. This robustness with respect to initial weight set is achieved through introduction of randomness in neuron weight space. Results are confirmed through experimental data and given in form of tables and graphs.

I. INTRODUCTION

Neural networks have been proven to be successful in many cases where other traditional techniques failed to give satisfactory results. Unfortunately, every architecture is not necessarily suitable for every given problem. Even with the correct architecture parameters, frustrating problems of correct choice of initial weights for certain architecture remain. These problems very often led to an unearned disbelief in neural networks as a technology.

This disbelief often comes from inexperienced users that do not have a "feeling" of adequate choice of network architecture and its parameters. This paper proposes an approach that would enable everyday user to learn, apply and gain experience with neural networks without a special expertise in this field. The suitability of this approach for everyday use comes from two reasons.

First reason for the suitability of the proposed approach for the inexperienced user is the accessibility of the application that is a web based. Application web interface is implemented in programming language Perl. For faster response, computationally intensive number crunching is written in C. Interface is therefore a web page that can be run independently of hardware platform.

Web interface, independent of hardware platform, software, operating system, compiler, etc. requires only Internet browser of any kind. In this way a transparent accessibility has been achieved from any kind of computer platform. This has been the reason of wide acceptance of engineering tools based on Internet.

Second reason is the methodology itself, which is not sensitive to initial choice of network parameters. This paper proposes an approach that automatically searches for an adequate initial weight set. This robustness comes with a price in computation time though. On the other hand, with the constant improvement of today computer computational power this issue becomes overcome.

Proposed method is computationally intensive. This is due to a large multidimensionality of weight set for any given network architecture. To illustrate this, consider the typical simple XOR example with architecture of three neurons on Fig. 1. Even in such elementary problem we have to deal with 9-dimensional space of neuron weights.

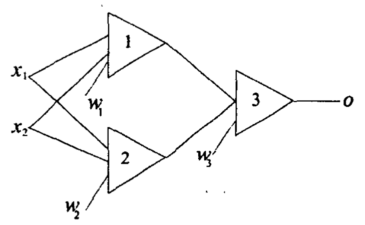

Fig. 1. Simple example of 3 neurons architecture.

The important issue of speed has been addressed in following manner. Only an interface has been done in interpreter (Perl) while the number crunching is done in C. An application gives an option of selective printing of intermediary results. If no intermediary results are transmitted execution time is equal time to send data + time to receive result + computational time. Execution time therefore is extended only for first transmission and final display of results. After all, the execution time is not crucial in case of neural network training, which brings us to a second reason.

Another benefit of the proposed application is that it runs through an Internet. By running on server machine where application resides, the burden has been taken off the client machine.

The rest of this paper is structured as follows. The overview of related work is given through the rest of this section. The second section proposes new approach. The third section provides the test example results. The fourth section concludes this paper with directives for future work.

Error Back Propagation (EBP) neural networks and gradient methods generally provide very good results. However, certain restrictions apply. In cases of large or small learning rate, process can either zigzag while going along the error gradient or can get stuck in local minima. Different heuristic approaches can be found in literature with variable learning rates. These methods are cumulative versus incremental weight updating, Sejnowski's momentum method and others [1]-[3]. Random learning constant values often lead to better results.

Different combined approaches of neural networks and genetic algorithms (GANN) can be found in literature. First pioneer attempts occurred in late 1980's [4]-[9]. Reviews can be found in [9]-[14].

Synergistic approach has been also applied to problems in medicine, such as neuromuscular disorders [16] or virtual design of pharmaceuticals [17]. Tewari et. al. addressed one of very often used applications - the estimation of the likelihood of cancer recurrence after surgery [18]. The same paper gives a thorough survey of similar work done.

Also one of appealing topics that could be found in literature are environmental resource management of freshwater and fisheries [19], hydroinformatics and rainfall runoff modeling [20]-[22]. Applications also include power systems [23], construction forecast [24], robotics [13]-[15].

Finally, probably the most common work done in a field of combined neural networks and genetic algorithms refers to an application of variety of evolutionary algorithms to automatic design of neural networks [9]-[13], [25]-[31]. Those attempts were driven by an important consideration of proper selection of network parameters, such as the number of hidden neurons and training parameters like learning rate and momentum factor. The determination of these parameters usually is done by means of heuristics or trial and error. There is no strict or reliable method developed for the determination of these parameters which is very necessary to get accurate network outputs [23].

II. PROPOSED METHODOLOGY

The proposed method combines approaches used in evolutionary computation and gradient based networks.

Genetic algorithms on one hand deal with reproduction with crossover and mutation always involving certain randomness. The advantage of evolutionary approach is the randomness that always leads to certain solution. Unfortunately, that solution does not necessarily have to be a satisfactory solution [10]-[12].

On the other hand, gradient neural network approach leads to a perfect matching. However, flat spot and problems with local minima with the gradient approach are always lurking around.

The proposed method encompasses selected features of these two techniques. Those are randomness that overcomes the problem of local minima and gradient approach that eventually leads to a solution of wanted accuracy.

Error Back Propagation (EBP) combines both forward and backward pass while training network to each pattern. Method proposed in this paper has no back propagation feature. Instead it performs single step forward propagation.

The selection of "winning" weight sets is performed based on criterion of minimal total error. Total error is the sum of errors for each pattern as in (1). Each error is difference between desired and calculated output.

$$TE = \sum_{p=1}^{P}(dout_p - out_p) \qquad (1)$$

The weighted average of those selected weight sets is calculated, and that point becomes the center of gravity around which a new "cloud" of random weight sets is generated. In next cycle, process does the same procedure - calculates total errors in single forward propagation cycle for each of the randomly generated weight matrices in that cloud.

Figures 2. and 3. show the application web interface and the results as a response to given inputs. The URL of an application is: http://husky.engboi.uidaho.edu/nn/.

The proposed algorithm goes as follows.

1. Generate a random set of weights that results in $n \times m$ array where n is number of neurons, and m is the number of weights of those neurons. A pool of such weight sets is generated (each weight array is therefore a point in multidimensional space).

2. Perform the forward propagation step i.e. calculate net, out values, and finally a network total error.

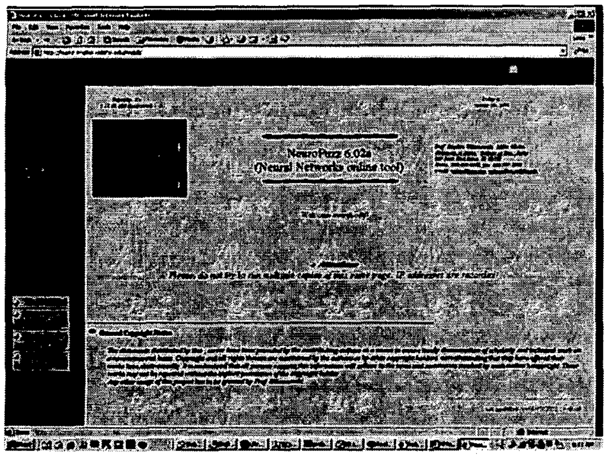

Fig. 2. Application's initial web interface.

This is performed for each randomly chosen weight set and for all patterns.

3. Check if total error fulfills the upfront given accuracy criterion. If so algorithm, has finished. Exit. If not, go to a next step.

4. In this step new center of gravity (COG) of the population pool is to be determined. First the selection of 10% of "winning" weight sets is done. Winners are weight sets with smallest total error. This percentage is experimentally adopted. This also means that the number of random generations in on cycle can not be smaller than 10 (in which case only one generation is selected to be carried over into next cycle). Now weighted average of those weight sets is to be calculated, and that point (averaged weight set) will become the new center of gravity.

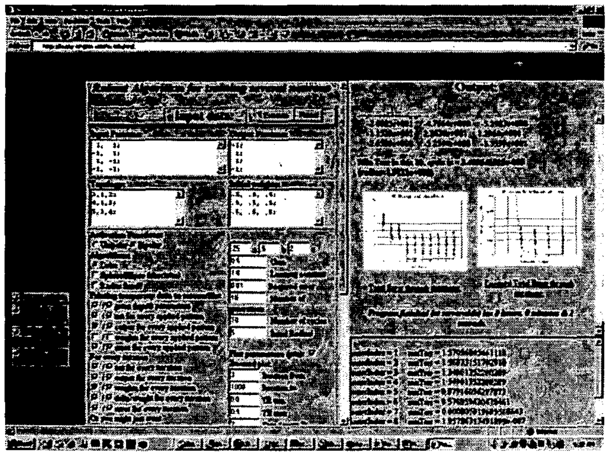

Fig. 3. Results that application returns in response to given inputs from previous Figure.

5. Weighted average is calculated in the following manner. 10% of total population is selected in calculation. Based on distance from a minimum achieved error in that set, weights to each of selected weight sets are assigned. Now this multidimensional point i.e. such calculated weight set is the next center of gravity around which a new "cloud" of random weight sets is going to be generated.

6. In case of first time passing through this step (no previous COG value), use from the interface predefined radius and go to step number 8. If not, calculate quasi-gradient based on previous and next center of gravity. In this way we try to do two things. First, check if the total error with respect to previous center of gravity is changing in satisfactory manner i.e. if we haven't got stuck in flat spot or local minima. Second, we try to predict future gradient change based on previous behavior.

7.
   - ✓ Decide whether new random generation cycle will be in shape of multisphere or multidimensional ellipsoid. In case of local minima or flat spot, increase the radius for random pool weight set generation to try to get out of local minima or flat spot. Keep the radius fixed (go with the sphere) to make an equal chance to all possible directions.
   - ✓ If quasi-gradient is satisfactory go with the ellipsoid. Accelerate the process by predicting the minima of total error parabola and update next COG. Set the radius now so the multidimensional ellipse towards gradient will be generated (instead of fixed radius - multisphere).

8. Now a new pool of weight sets is generated. Create new pool of randomly generated weight sets around a point obtained from previous step. While creating new pool, use decision regarding shape (multisphere or multidimensional ellipsoid) from previous step. Previously selected 10% of weight sets from the step 5 (sets with the smallest total errors) are kept. This is done for two reasons. First, it saves computational time and second, even more important is that in this manner we keep in population pool possible winners. These possible winners are kept because in following cycle they do not necessarily have to occur again.
9. Go to step 2.

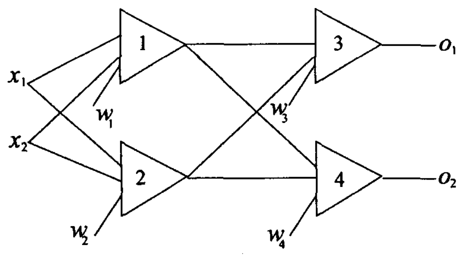

Fig. 4. Neural network architecture for the test example.

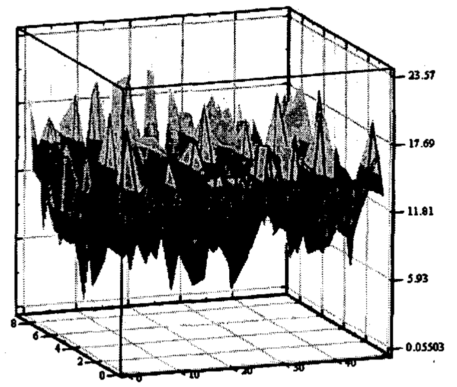

Fig. 5. Total error (TE) for the pool of randomly generated weight sets (TE for each weight set through cycles, 50 iterations per each pool generation cycle).

III. EXPERIMENTAL RESULTS

The test example used to verify the proposed methodology considered the problem with the network architecture from Fig. 4. Neural network parameters used are given in Table I. The overview of comparison of these two methodologies is given in Table II.

The purpose of the experimental results was not to give a comparison of proposed and EBP algorithm in terms of speed, number of iterations or accuracy. This is due to significant difference between steps in. However, certain conclusions were derived and given at the end of this section.

For the purpose of testing proposed methodology, results were obtained through 10 consecutive cycles of 50 random weight generation each. The network has 2 input and 2 output neurons. Parameters used in forward propagation cycle are for learning parameter alpha=1.0 and for gain k=0.5. Activation function was bipolar, rule used was Delta rule.

With these parameters EBP algorithm

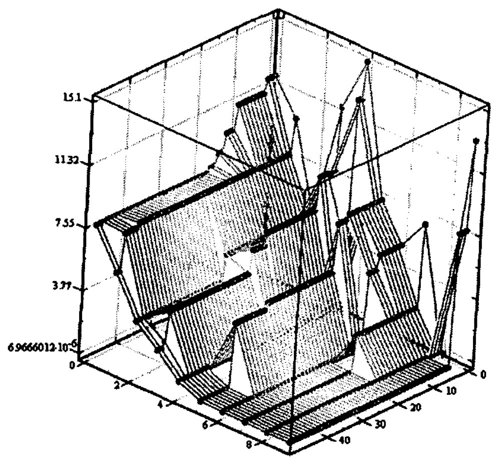

Fig. 6. Learning curve (TE) through 10 cycles, 50 generation for each cycle.

did not converge. It did however for smaller values of alpha and gain. One of solutions for EBP were achieved for values of learning parameter and gain 0.3 and 0.5, but in 5000 steps.

Total Error (TE) for the pool of randomly generated weight sets through 10 consecutive cycles is graphically represented by Fig. 5. Randomness is introduced upon given initial weight set starting from initial radius 5. Variation of randomly generated weights in interval $(-20 \div 20)$ can be noticed.

TABLE I
PATTERNS FOR THE FIRST TEST TRAINING EXAMPLE

Input patterns	Output patterns
+1, -1;	+1, -1;
+1, +1;	-1, +1;
-1, +1;	+1, -1;
-1, -1;	-1, +1;

TABLE II
COMPARISON OF EBP AND PROPOSED TECHNIQUE FOR THE FIRST TEST EXAMPLE

	No. of iterations	Total error	Convergence	Each iteration includes:
EBP (gain=0.3, alpha= 0.5)	5000	3.67 e - 03	Parameters have to be manually adjusted	Forward + backward pass
Proposed methodology	10 cycles, 50 gen.each	6.97e-06	Automatically adjusts until convergence	Simple forw. pass (TE) + rand. space generation

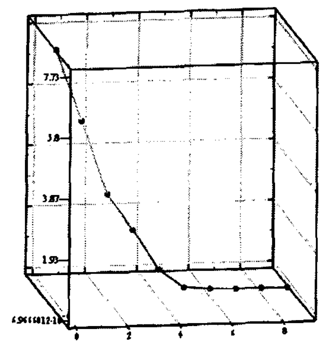

Fig. 7. Total error through cycles for the proposed technique

Figure 6. represents the change of minimal TE through cycles. For each cycle (1-10), from 1st to 50th generated weight set, decrease in TE value is obvious. After only a few first iterations, TE decreases from values in interval (8, 15). For the rest of around 45 iterations TE gets smaller values.

For each following cycle, the speed of TE decrease is larger. In each cycle, after only a couple of iterations, error drastically decreases. In last couple of cycles (9th to 10th), TE becomes zero after only a few iterations. Algorithm therefore in each cycle gets obviously closer to wanted minimal TE.

On Fig. 7. TE through cycles is given. TE values are in range of 7.73 to 6.97e-06 for cycles 1 to 10. This graphs shows exponential quasi-gradient behavior of TE that can be exploited for guessing of the final weight set.

Speed would be unrealistic to compare only with respect to number of iterations or cpu time. This is because each iteration involves different steps in these two algorithms. Also with respect to speed, EBP algorithm in each iteration performs both forward and backward propagation (weights adaptation). The methodology proposed in this paper performs only simple total error calculation, without its back propagation i.e. without weight update.

Proposed methodology in this example converged to a larger total error. Also, the price for robustness (automatic selection of weight values) has been paid in terms of processor time regarding generating large random population that are later on used for selecting winner points with minimum total error values.

Even though certain conclusions can be driven out of presented example, authors are inclined not to do that until more experiments are performed.

Finally, accuracy discussion between two methodologies was not the prime goal of this paper. Naturally, with both techniques, repeating the same procedure with different parameters gives quite different results but following conclusions can be derived.

On Fig. 8. the ratio of TE for random pool in sphere and ellipsoid shape is given. Abscissa gives negative exponent of TE, while ordinate shows the occurrence frequency of TE for in 100 times program run.

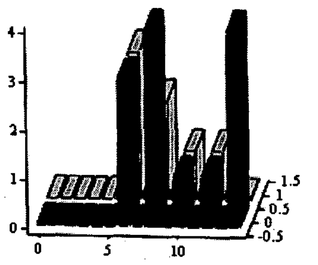

Fig. 8. TE for randomly generated in shape of multidimensional ellipsoid and multisphere, for the same example.

It can be seen that once the correct quasi-gradient direction is chosen, ellipsoid converges to smaller TE values (10^{16} or less). Ellipsoid modification accelerates convergence for about 35%.

Both methodologies can be automated for different parameters. Proposed one does not require any user experience with tuning of network parameters until final convergence. The less smooth the function to be learned is, the more this difference becomes obvious. The main advantage of the proposed methodology that has been proven is the robustness with respect to initial choice of network parameters, i.e. network weight set, learning, and gain parameter.

Through different runs of the same algorithm, due to its a random nature results varied. Still, in each case algorithm converged. Proposed methodology can be therefore efficiently exploited for experimenting and finding initial network architecture and parameters for the given problem. Upon that, other possibly more optimal solution can be manually tuned and other algorithms can be then applied.

IV. CONCLUSION

Numerous attempts to prove superiority of neural networks very often have often failed due to a problem of inadequate choice of weight set. Experienced users can "feel" the right choice for weight set, but problems still occur in everyday usage.

This paper proposes one solution for robustness of neural networks with respect to initial choice of weight set. Experimental results have proven the independence of convergence with respect to initial choice of weights. This comes with a price of computational time, though. However, the issue of speed is only important in applications for monitoring and control. These applications are not prime targets for implementing neural networks anyway. Obtained experimental results were represented by tables and figures.

The purpose of the experimental results was not to give a comparison of proposed and EBP algorithm in terms of speed, number of iterations or accuracy. This is due to significant difference between steps. However, the main advantage of the proposed methodology has been proven. That is the robustness of this methodology with respect to initial choice of network parameters, i.e. network weight set, learning, and gain parameter.

Further work includes different directions. In EBP networks, randomness can be introduced in weight set as well. Through a traditional EBP process, in cases when network runs into problems of local minima, randomness in weight set can be introduced. Also, change of total network error through introduction of randomness can be investigated in terms of speeding up the process, i.e. predicting final solution after two or more steps.

V. REFERENCES

[1] J.M. Zurada, *Artificial Neural Systems*, PWS Publishing Company, St. Paul, MN, 1995.

[2] [Hagan 96] M.T. Hagan, H.B. Demuth, M.Beale, *Neural Nework Design*, PWS Co., Boston MA, 1996.

[3] R.J. Schalkoff, *Artificial Neural Networks*, McGraw-Hill, New York, 1997.

[4] W.B. Dress, "*Darwinian optimization of synthetic neural systems*", San Diego, CA:SOS, 1987.

[5] A. Guha, S. Harp, T. Samad, "*Genetic synthesis of neural networks*", Honeywell Corporate System Development Division, Tech. Rep CSDD-88-I4852-CC-1, 1988.

[6] D. Whitely, "*Applying genetic algorithms to neural net learning*", CS Department, Colorado State Univ., Tech Rep. CS-88-128, 1988.

[7] G.F. Miller , P.M. Todd, S.U.Hedge, "*Designing neural networks using genetic algorithms*", In Proc. 3rd Int. Conf. Genetic Algorithms, 1989.

[8] R.K. Belew, J.McInereny, N.N. Schraudolph, "*Evolving networks: Using the genetic algorithm with connectionist learning*", CSE, University California San Diego, Tech. Rep. CS90-174, 1990.

[9] J.D. Scahffer, D. Whitley, L.J. Eshelman, "*Combinations of genetic algorithms and neural networks: A Survey of the state of the art*", In Combinations of Genetic Algorithms and Neural Networks, IEEE Computer Society Press, Los Alamitos, CA, pp. 1-37, 1992.

[10] I. Kuscu , C. Thorton, "*Design of artificial neural networks using genetic algorithms: Review and prospect*", University Sussex, Brighton, U.K., Cognitive Sci. Res. Paper 319, 1994.

[11] K. Balakrishnan , V. Honavar, "*Evolutionary design of neural architectures – preliminary taxonomy and guide to literature*", Artificial Intelligence Group, Iowa State University, Ames, Tech. Rep. CS TR#98-01, 1995.

[12] J. Branke , "*Evolutionary algorithms for neural network design and training*", In Proc. 1st Nordic Workshop Genetic Algorithms Applications, T. Talander, ed., 1995.

[13] J. Kodjabachian, J.A. Meyer, "*Evolution and development of control architectures in animats*", Robot. Autonomous. Systems, vol 16, pp.161-182, 1995.

[14] T. Gomi , A. Griffith, "*Evolutionary Robotics – an overview*", Proc. IEEE 3rd Int. Conf. Evolutionary Computation, 1996.

[15] J. Kodjabachian, J.A. Meyer, "*Evolution and Development of neural controllers for locomotion, gradient following and obstacle avoidance in artificial insects*", IEEE Transactions on Neural Networks, Vol. 9, No.5, pp.796-812, 1998.

[16] C.S. Pattichis, C.N. Schizas, "*Genetics-Based Machine Learning for the Assessment of Certain Neuromuscular Disorders*", IEEE Transactions on Neural Networks, Vol. 7, No.2, pp. 427-439, 1996.

[17] R.H. Kewley, M.J. Embrechts, "*Data Strip Mining for the Vritual Design of Pharmaceuticals with Neural Networks*", IEEE Transactions on Neural Networks, Vol. 11, No.3, pp. 668-679, 2000.

[18] A. Tewari et. al., "*Genetic Adaptive Neural Network to Predict Biochemical Failure After Radical Prostatectomy: A Multi-institutional Study*", Molecular Urology, Vol. 5, No.4, pp. 163-169, 2001.

[19] O. Varis, "*Bayesian decision analysis for environmental and resource management*", Environmental Modeling Software 12, pp.177, 1997.

[20] V. Babovic, Q.W. Minns, "*Use of computational adaptive methodologies in hydroinformatics*", Proceedings of the First International Conference on Hydroinformatics, international Institute for Infrastructural, Hydraulic and Environmental Engineering, Delft, The Netherlands, pp. 201-210, 1994.

[21] V. Babovic, Z. WU, L. Larsen, "*Calibrating hydrodynamic models by means of simulated evolution*", Proceedings of the First International Conference on Hydroinformatics, international Institute for Infrastructural, Hydraulic and Environmental Engineering, Delft, The Netherlands, pp. 193-200, 1994.

[22] J.P. Drecqurt, "*Application of Neural Networks and Genetic Programming To Rainfall Runoff Modeling*", D2K Technical Report (D2K-0699-1), Danish Hydraulic Institute (Hydroinformatics Technologies, HIT).

[23] A. Mohamed , S. Maniruzzaman, A. Hussain, "*Static Security assessment of a power system using genetic based neural network*", Electric Power Components and Systems, 29, pp.1111-1121, 2001.

[24] G.B. Hua , "*Evaluating the performance of combining neural networks and genetic algorithms for forecast construction demand: the case of the Singapore residential sector*", Construction Management and Economics 18, pp.209-217, 2000.

[25] V. Maniezzo, "*Genetic evolution of the topology and weight distribution of neural networks*", IEEE Transactions on Neural Networks, Vol. 5, No.1, pp.39-53, 1994.

[26] P.J. Angeline , G.M. Saunders, J.B. Pollack, "*An evolutionary algorithm that constructs recurrent neural networks*", IEEE Transactions on Neural Networks, Vol. 5, No.1, pp.54-65, 1994.

[27] R.A.Zitar, M.H.Hassoun, "*Neurocontrollers Trained with rules extracted by a genetic assisted reinforcement learning system*", IEEE Transactions on Neural Networks, Vol. 6, No.4, pp. 859-879, 1995.

[28] S. Park, L.J. Park, C.H. Park, "*A neuro-genetic controller for nonminimum phase systems*", IEEE Transactions on Neural Networks, Vol. 6, No.5, pp. 1297-1300, 1995.

[29] Y. Matsuyama, "*Harmonic Competition: A self-organizing multiple criteria optimization*", IEEE Transactions on Neural Networks, Vol. 7, No.3, pp. 652-668, 1996.

[30] S. Mizuta , T. Sato, D. Lao, M. Ikeda, T. Shimizu, "*Structure design of neural networks using genetic algorithms*", Complex Systems, 13, pp. 161-175, 2001.

[31] G.G. Yen, P. Meesad, "*Constructing a fuzzy rule based system using the ILFN network and genetic algorithm*", Int. Journal of Neural Systems, Vol.11, No. 5, pp.427-443, 2001.