Nonlinear Dynamics and Systems Theory, **10** (3) (2010) 269–282

A Hierarchical Genetic Algorithm Coding for Constructing and Learning an Optimal Neural Network

Imen Ben Omrane* and A. Chatti

Institut National des Sciences Appliquees et de Technologie INSAT, Centre Urbain Nord BP 676 - 1080 Tunis Cedex, Tunisie

Received: April 28, 2009; Revised: July 17, 2010

Abstract: Neural Networks (NN) proved to be a powerful problem solving mechanism with great ability to learn. The success and speed of training is based on the initial parameter settings such as architecture, initial weights, learning rates and others. The most used method of training Neuron Networks is the back propagation of the gradient. Although this method provides a global optimal solution in a reasonable time, it can converge towards local minimum, in addition to large number of parameters that should be fixed previously. Within this framework of study, we propose a new coding for a hierarchical genetic algorithm for the determination of the structure and the training of the Neuron Networks. These algorithms are known for structures' and parameters' optimization. We will prove that Hierarchical genetic algorithm can improve the result of backpropagation of gradient.

Keywords: *hierarchical genetic algorithms, neural networks, backpropagation algorithm, learning, multilayer perceptron, optimization.*

Mathematics Subject Classification (2000): 93A13, 92B20.

1 Introduction

The solutions to the problem of learning neural networks by back propagation of gradient are characterized by their incapacity to escape from local optima. The evolutionary algorithms bring a great number of solutions in certain fields: drive networks of variable architecture, automatic generation of Boolean Neurons Networks for the resolution of a certain class of problems of optimization [15]. But the research effort is especially related to the generation and the training of discrete networks.

* Corresponding author: mailto:imenbo7@yahoo.fr

In this paper, we propose a solution of research for the learning and the determination of Neural Network structure. This solution is based on the hierarchical genetic algorithm which can escape local optima. The evolutionary algorithms are based on the study of the process of natural evolution. The important principles of this type of algorithms are the following:

- The algorithm works on a population of individuals. Each individual corresponds to a point of research on a space of solutions.
- The population is initialized by chance. It evolves/moves thanks to operations such as the change of an individual or the recombination of individuals.
- The adaptation of an individual to his environment is measured thanks to a function called Cost which associates a positive reality with each individual.

Genetic Algorithms (GA) form a subset of the evolutionary algorithms. An individual is entirely determined by a genotype (chromosome), a phenotype and a reality [12]. By analogy with the process of natural evolution, a chromosome is often a chain of bits or integer. The operations of change and recombination are carried out by specific operators. Empirical studies showed that GA converged towards a total optimum for a big class of problems of optimization defined for a discrete unit [9]. Theoretical studies demonstrated that under certain hypothesis of regularity for the force function, these algorithms converge asymptotically into total optimal solutions [9].

We will present in this work a Hierarchical Genetic Algorithm (HGA), which is an alternative to the Genetic Algorithms. HGA has as role to optimize parameters and the structure (number of neurons in hidden layer, input and output weights), and at the same time, to train the network for modelling a non-linear process. Algorithms will do a research in a very varied environment, in order to explore all the possible solutions and to avoid the local minima or to be able to leave them.

To summarize, our work consists in minimizing the function cost (the quadratic error) into two steps: an algorithm of back propagation of gradient starting the minimization, then a HGA continues the work by carrying out research for another architecture in order to leave the local minima and to find the global one.

Two examples will be employed to proof the improvement of the HGA solution compared by methods of gradient.

2 Learning Process

Once the architecture of a NN is selected, it is necessary to do a learning to determine the values of the weights making the NN output to be as close as possible to the laid down objective. This learning is carried out thanks to the minimization of a function, called function cost, calculated from the examples of the training and the output of the NN; this function determines the objective to reach.

2.1 Principle of the minimization algorithms

The principle of these methods is to use an initial point, to find a direction of descent of the cost in the space of the parameters W, then to move a step in this direction. Once a new point is reached, we reiterate the procedure until satisfaction of a criterion of stop. Thus, in the kth iteration, we calculate $\omega_k = \omega_{k-1} + \alpha_{k-1}.d_{k-1}$, where α_k is the step of the descent and d_k is the direction of descent. Various algorithms are characterized by the choice of these two parameters.

2.2 Problem of the local minima

The minimum found by the back propagation algorithms is local minima. The minimum found depends on the starting point of the research i.e. of the initialization of the weights. In practice, it is necessary to carry out several minimizations with different initializations, to find several minima and to keep the "best". It is nevertheless impossible and generally useless to make sure that the selected minimum is the global one.

3 Presentation of the Hierarchical Genetic Algorithms (HGA)

3.1 Principle

Contrary to the conventional genetic algorithms, the Hierarchical Genetic Algorithms (HGA) use variable chromosomes dimensions. These chromosomes have the specific name of Hierarchical Chromosomes (HC). They are used mainly for the joint optimization of structure and parameters. The principle of HC is the following: the activation of a parametric gene is guided by the value of a gene of control of the first level, which is activated by a gene of control of the second level, etc [7]. Figure 1 illustrates this mechanism.

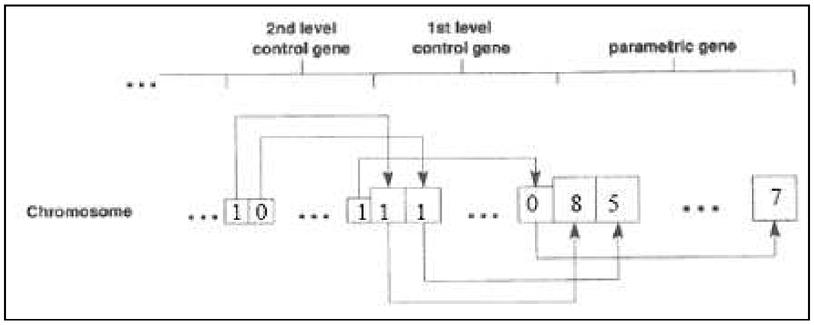


Figure 1: Structure of a Hierarchical Chromosome.

The HGA have a flexible hierarchical structure. Indeed, several hierarchical levels can be used if needed. Therefore it is a question of an ideal approach to model topologies or structures at the same time as the corresponding parameters. The HGA will seek a solution considering all the possible lengths (different structures) and all the values of parameters to meet the criteria of the objective function.

3.2 Genetic operations on the Hierarchical Chromosomes

Since the structure of the Hierarchical Chromosomes is fixed, the methods of crossing and mutation can be used independently for each level or on the entire chromosome if it is homogeneous. The genetic operations which affect genes of the higher levels can affect the number of active genes of the lower levels what makes it possible to jointly optimize the parameters and the topology of the system to be optimized [14].

3.3 Multi-criterion approach

Let us define the objective functions F associated with each chromosome X of a HGA: $F = [f_1 f_2 ... f_i]'$. The objective functions F are used by the HGA for the optimization of the parameters of the system. The task of a HGA is the joint optimization of the parameters and the topology of a system, so an additional objective function f_{i+1} must be considered by the HGA for the optimization of the topology of the system (Figure 2). Based on the specific structure of the chromosome of the HGA, information concerning topology to be optimized is directly acquired by genes of control.

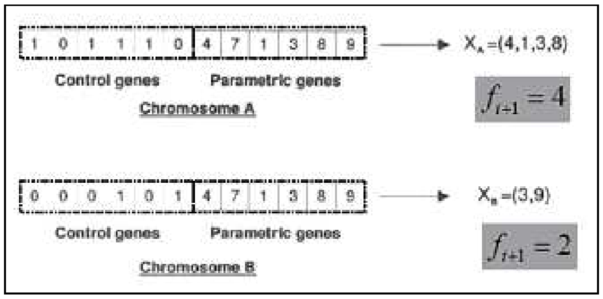


Figure 2: Example of optimization of the structure.

By regarding topology as an objective to be optimized, the problem is now formulated like a problem of multi-criterion optimization $F = \begin{bmatrix} F_i \\ f_{i+1} \end{bmatrix}$.

The genetic algorithm is used to minimize the vector of objective functions F.

Let us consider a problem of a system optimization, where:

- x_j represents a solution (chromosome),
- $F_i(x_j) = M_j$ represents the objective function to minimize.
- $f_{i+1}(x_j) = n_j$ represents the number of parameters of the system.

The whole of the acceptable solutions to this problem is represented by: $\{x : F_i(x) = 0 \text{ at } N1 \leq f_{i+1} \leq N2\}$. In short, the best solution to this problem is

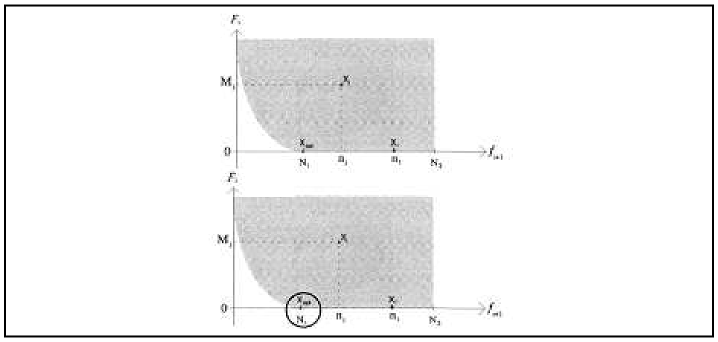


Figure 3: Whole of the acceptable solutions for a problem of joint optimization of the parameters and topology of a system.

represented by x_{opt}, where $f_{i+1}(x_{opt}) = N1$ and $F_i(x_{opt}) = 0$.

Other solutions where the value of the objective function $f_{i+1}(x_j) = n_j > N1$ with $F_i(x_j) = 0$ are acceptable solutions as regards the parameters of the system however the topology of the system is not optimal (Figure 3) [14].

4 HGA for NN learning

HGAs are used for the optimization of the parameters and the topology of NNs. The advantage of this approach is that the genes of the chromosome are classified in two categories (hierarchy). This approach is ideal to represent the relations between:

- layers of the network,
- neurons in hidden layer,
- weights.

The following Figure 4 illustrates the principle operation of the new strategy.

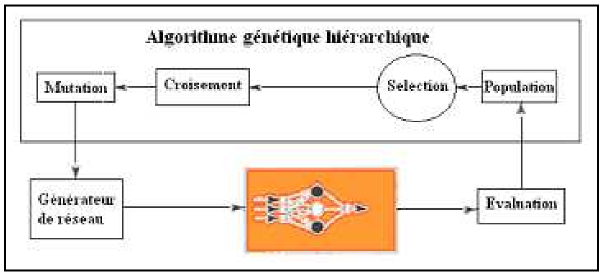


Figure 4: Principle of the new strategy.

Each chromosome is composed by 2 types of genes:

- Genes of control (bits) for activations of the layers and the neurons of the NN.
- Genes of connections (real) for the determination of the weights of connexion.

The following Figure 5 illustrates the Structure of a chromosome.

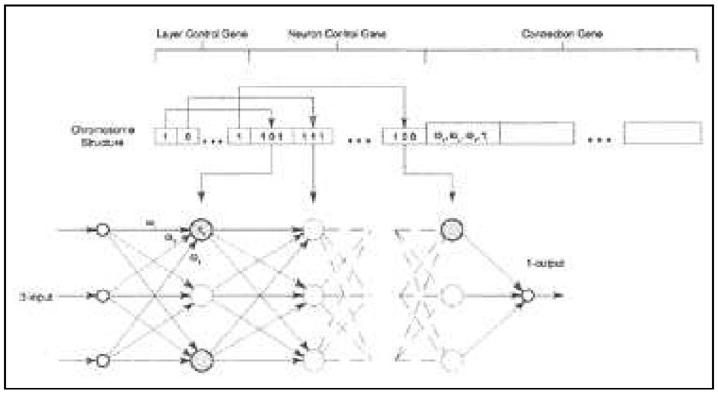


Figure 5: Structure of a chromosome for the optimization of a NN.

5 Presentation of New Approach for NN Learning

5.1 Objective

We build a HGA which selects a structure of NN (number of neurons in the hidden layer). This number is given after the evaluation of a cost function. The function to be minimized is the following quadratic criterion:

$$J = \frac{1}{N}\sum_{k=1}^{N}[y_{di} - y_i]^2, \tag{1}$$

where N is a number of example, y_{di} is an output of the process, y_i is an NN output.

5.2 Coding

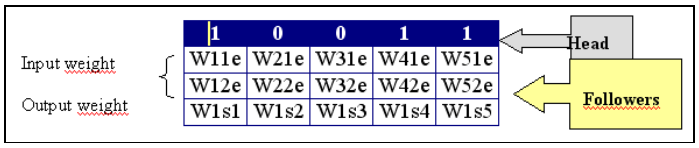
We have N_{pop} chromosomes which are in the form of multi-dimensional table. The first line is coded with 0 and 1 which indicates the consideration or not of neuron in hidden layer. The lines which remain contain real which represent the whole of connexions of input and output of hidden layer.

Example of chromosome coding:
We consider a NN with 2 input layer, 1 output layer and maximum 5 neurons in hidden layer (this number can change).

Thus, our chromosome will have 5 columns and 4 lines:

- The columns represent neurons constituting the hidden layer,
- The 1st line contains 0 and 1. If 1 then the neuron exists (active) if 0 then it's not consider (inactive). In this example a hidden layer contains 3 neurons (There are 3 box with 1),
- 2nd and 3rd line represents input weights (2neurons of entry thus 2 lines)
- 4th line represents output weights (1 neuron of exit thus a line)

Figure 6 illustrates the description bellow.



1	0	0	1	1
W11e	W21e	W31e	W41e	W51e
W12e	W22e	W32e	W42e	W52e
W1s1	W1s2	W1s3	W1s4	W1s5

Input weight
Output weight
Head
Followers

Figure 6: Chromosome code.

The corresponding NN is:

5.3 Evaluation

Feval represents the quadratic criterion to minimize. Each individual of the population is evaluated independently of the others

$$F_{eval}(x) = \frac{1}{N}\sum_{k=1}^{N}[y_{di} - y_i]^2. \tag{2}$$

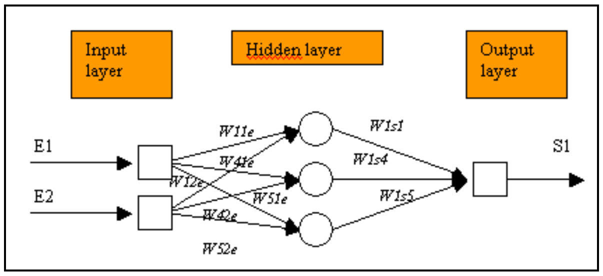


Figure 7: translate chromosome to RNA.

The function of evaluation is a function which depends on 2 independent variables : the first parameter is the desired output y_{di}. This output, we obtained it after having applied to our process an entry rich in frequency (learning sequence). The second variables y_i are the output which are calculated as follows:

$$y_k(\boldsymbol{x}) = h\left(\sum_{k=0}^{M} w_{ki}^{(2)} g\left(\sum_{j=0}^{D} w_{ij}^{(1)} x_j\right)\right)$$

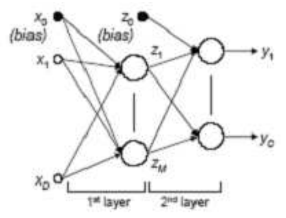


Figure 8: Calculation of the exit of the network.

5.4 Initial population

It is necessary to create and maintain a sufficient genetic diversity in the population. For this reason the initial population must be most heterogeneous as possible in order to prevent premature convergence. That is why we chose a random generation of N_{pop} individuals of the initial population. But for not having, a research space far from the required minimum, and to minimize the search time, we chose to inject into this initial population a certain number of individuals resulting from a training by back propagation.

5.5 Selection

We select a group of reproducers according to their evaluation by the cost function Feval. The individuals who have the minimal function will be selected. It is about an elitist selection. It was noted that this method induced a perfect convergence of the algorithm. Indeed, any Good individual which once combined with others also good, can appear to be interesting. For each individual:

- Calculate the function cost relative to each individual,
- Sort individuals in the ascending order of their function cost,
- Copy in the following generation the Pselect better individuals.

5.6 Criterion of stop

The iterations of the algorithm end when the maximum number of generation, defined in advance, is reached.

6 Application

6.1 System 1: Simple pendulum

We consider the non-linear system describing a simple pendulum and represented by the following model:

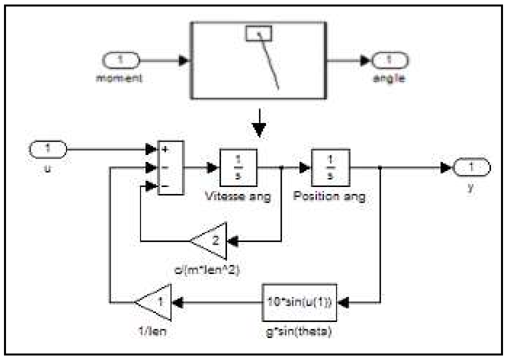


Figure 9: Model of the simple pendulum.

We propose to apply for this system the neuronal predictive control in order to control the position.

We will proceed by creating examples of learning. Then we will identify direct neuronal model of the system by two methods of training: backpropagation and HGA [4, 5].

Development of the learning examples.

These examples of learning constitute a database which will be used for the identification of the direct neuronal model by training. By considering a field of study, we excite the system with input vector (which is a control signal) of amplitude between [-10, 10] and we recover the corresponding vector which is in the interval [-1, 1]. We divide this database into two parts: One will be used for the learning and the other for validation. Figure 10 shows the sequence of test used.

Development of direct neuronal model: DNM :

Establish the system model to be used later in system control. We will follow the choice below:

- Choice of the representation: input/output,
- Choice of the model assumption: model NARX (model not looped),
- Choice of the order of the model: 2.

By considering these choices, the network predictor is governed by the following equation: $yr(k) = \varphi(y(k-1), y(k-2), u(k), u(k-2); C)$, where yr is NN output, y is a system output, φ is a function of network, C is network parameters (weight).

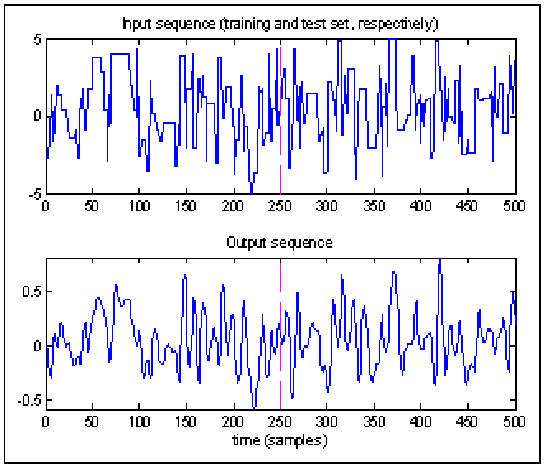


Figure 10: Input/output vectors (sequences of training and of test).

Architecture of the network.

We choose network not looped with 1 hidden layer: 4 inputs, 10 neurons hidden which have a sigmoid function (tansig) and 1 neuron of linear output. The input of the network is the vector: $Ed = [y(k-1); y(k-2); u(k-1); u(k-2)]$.

Consider the matrix of the input weights W_d and the matrix of the output weights W_{ds}. W_d is a matrix (10,4), W_{ds} is a matrix (1,10), ym is a network output, such as $ym(k) = W_{ds}$*tansig$(W_d * Ed)$;

Learning.

Used algorithm of training is the back propagation of the simple gradient to minimize the following criterion:

$$J = \frac{1}{2}\sum_{k=1}^{N}[y(k) - y_m(k)]^2, \tag{3}$$

where J is a quadratic error, N is a number of learning examples.

We use the following parameters: the iteration $N = 200$; a step of training $= 0.2$; values of initial input $y(1) = 0, u(1) = 0$; and a random initialization of the matrices of the weights.

Wd = Rand (10,4) = [0,52952 0,081759 -0,4405 -0,55428 -0,97313 0,63588 -0,020734 -0,028146 0,47375 0,40174 0,024 0,017082 -0,0060763 0,23539 -0,51866 0,32817 0,75842 -0,50884 0,0193 0,037063 0,25786 -0,08087 -0,25978 -0,25763 0,043251 0,44786 -0,42815 0,34426 -1,4512 0,69197 -0,021571 0,020547 0,21106 -0,073689 -0,28491 -0,38401 -0,13845 0,58937 -0,088476 -0,09446],

Wds = Rand (1,10) = [0,086723 -0,32983 0,17984 -0,068595 0,2788 0,015448 0,018001 -0,344 -0,072587].

Then we use the algorithm of back propagation for a given iteration.

Test of the model.

To estimate the error of generalization by calculating the average quadratic error on the sequence of test and of training according to the following formula:

$$EQMT = \sqrt{\frac{1}{N_T}\sum_{k=1}^{N_T}[y(k) - y_m(k)]}, \tag{4}$$

$$EQMA = \sqrt{\frac{1}{N_A}\sum_{k=1}^{N_A}[y(k) - y_m(k)]}. \tag{5}$$

The validation of the model found is done by applying to the NN input, after learning, test sequence and by comparing the output with the output wished. We evaluate the performance of the model found by the error analysis of prediction which represents the error of generalization of the network (Figure 11) [4, 5].

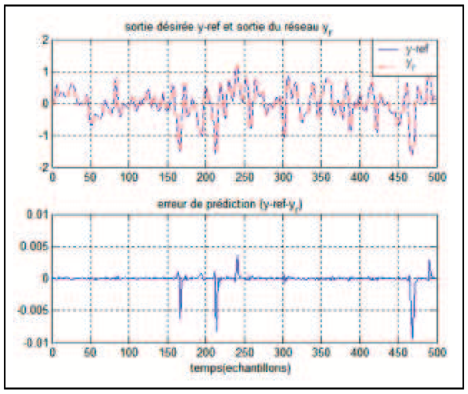


Figure 11: NN output and error of prediction of the DNM.

First, we made the learning of NN with the back propagation algorithm of the gradient. Now, we apply our HGA for determining the NN architecture and make learning. The parameters of simulation which we used are the following:

- The number of maximum neuron that our network can contain is 20 neurons.
- The number of neuron in the input layer is 4 (Ed=[y(k-1); y(k-2);u(k-1) ; u(k-2)]).
- The number of neuron in the output layer is 1.
- The number of individuals in a generation is 200.
- The number of generation (iteration) is 100.

After simulation, the algorithm gives us, a NN of 9 neurons in the hidden layer and the following matrices of weight:

Input Weight Wd: [0,52879 0,081639 -0,55371 -0,44001 -0,96342 0,62766 -0,031226 -0,023709 0,46174 0,40518 0,027177 0,020853 0,74751 -0,5037 0,021788 0,039943 0,25955 -0,079655 -0,25648 -0,24315 -1,4033 0,65602 -0,022435 0,014063 -0,10963 0,55455 -0,098256 -0,099649 -0,56835 1,0888 0,56752 0,020133 -0,074531 -0,87235 0,63455 0,3784].

Output Weight Wds:[0,081874 -0,33508 0,17908 0,28084 0,0172 -0,39829 -0,22687 -0,016373 0,008739].

Now we will evaluate the performance of the model found by the error analysis of prediction which represents the error of generalization of the network (see Figure 12).

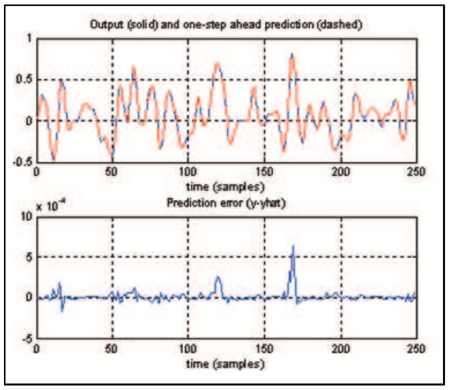


Figure 12: NN output and prediction error of the DNM after use of the HGA.

✓**Comments.**

According to Figure 12, we notice that the output of the network reproduces the evolutions of the output of the process. This results in an error of prediction very close to zero which reaches the maximum of 0.0005. So, the model established by the HGA gives a better result than that found previously by back propagation of the gradient. We can say that the solution given by back propagation of the gradient was local minima. HGA find another best minimum by using another structure and parameters of NN.

6.2 System 2: Rigid spring shock absorber

We consider the non-linear system describing a rigid spring shock absorber. It is represented by the following differential equation :

$$\ddot{y}(t) + \dot{y}(t) + y(t) + y^3(t) = u(t). \tag{6}$$

Figure 13 represents block of our system where y is a system output representing the linear position, u is a system input representing the force applied: y is a system output

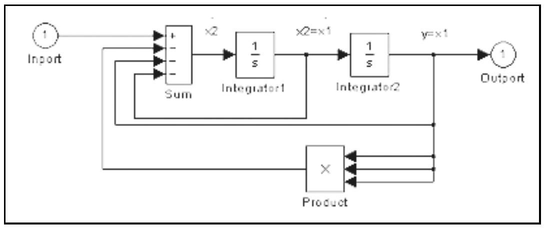


Figure 13: Model of spring shock absorber rigid.

representing the linear position, u is a system input representing the force applied.

Development of the examples of learning.

We recover the examples of learning by exciting the system with an input vector of frequency and amplitude variable between [-12, 12]. We recover the corresponding output vector (Figure 14). This database will be used for the identification of direct model MND and inverse model MNI of the process. We divide the vectors of input and output into two parts of which one will be used for the learning and the other for validation.

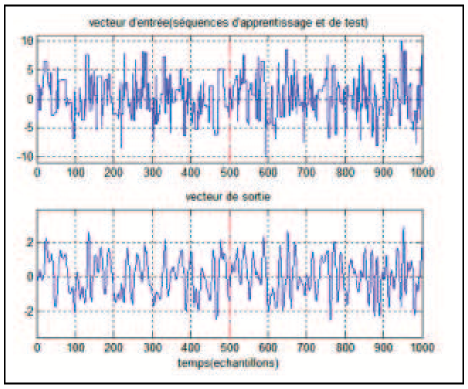


Figure 14: Input/Output Vectors (sequences of training and of test).

Inverse Model Control (IMC).

We propose to apply inverse model. We made learning of inverse model of the process by two methods which will be compared. The first one by direct learning and the other by HGA learning. Then, we will simulate the two inverse model control found in order to compare the performances of the two types of learning.

Learning direct inverse model.

Learning consists in minimizing the following criterion:

$$J = \frac{1}{2}\sum_{k=1}^{N}[u(k) - u_m(k)]^2, \tag{7}$$

where J is a quadratic error, N is a number of training examples.

The characteristics of the adopted network are the following ones:

- NN input : process order is 2 so the regressor is $(y(k+1), y(k-1), u(k-1), u(k-2))$,
- Network architecture: 5 neurons in hidden layer with sigmode activation functions, one linear output neuron.

Validation of the IMC.

We validate the model found after its learning by applying to the network the sequence of test and while comparing with the desired output. Figure 15 shows results found in order to evaluate the performance of the NN model [4, 5].

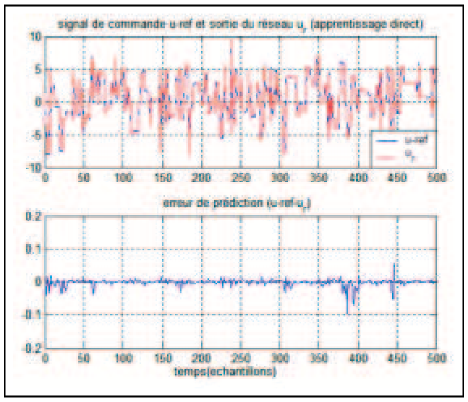


Figure 15: NN output and error of prediction of the IMC (direct learning).

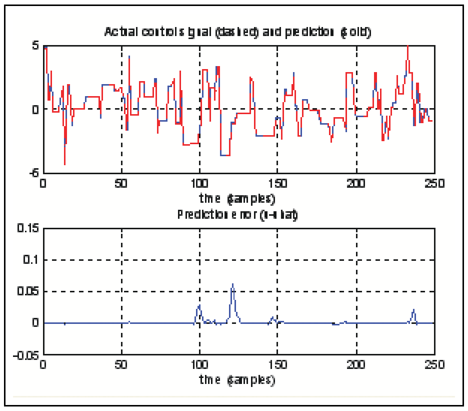


Figure 16: NN output and error of prediction of the IMC (direct learning) after use of AGH.

Now, we apply our HGA for learning inverse model. For the validation, we apply to the network found the sequence of test and we will compare it with the desired output. Figure 16 shows the results found and error of prediction in order to evaluate the performance of the network model.

✓**Comments.**

We note that the IMC found after the use of our hierarchical genetic algorithm is very satisfactory compared to the traditional method used at the beginning. Indeed we have an error of prediction much nearer to 0.

7 Conclusion

The algorithms performing local searches based on gradient information are sensitive to initialization and can provide a solution corresponding to a local optimum. This sensitivity to local optima is very limiting for multilayer perceptron learning.

Our main aim is the limitation of the defects of back propagation algorithm. The HGA gives us a very interesting result. Indeed this algorithm considered very great number of multi-layer architecture of perceptron and made a learning by applying its various operators (crossover and mutation).

In order to test our algorithm, precisely effectiveness of coding, we simulated two non-linear systems, by the Simulink tool of MatLab. We have initially made learning of direct and inverse model by using backpropagation and then our HGA. The results show that the variance of the error for the second method is better than for the first one and can escape local optima.

References

[1] Borne, P., Benrejeb, M. and Haggege, J. *Les rseaux de neurone- Prsentation et application.* TECHNIP, 2008.

[2] Bishop, C.M. *Neural Networks for pattern Recognition.* Clarendon Press, Oxford, 1995.

[3] Chatti, A. and Ayari, I. Identification and Control of a Non-Linear Process using a neural network approach. In: *Congrs International en Sciences et Techniques de lAutomatique STA*, Sousse, dcembre, 2005.

[4] Chatti, A., Gallah, T. and Ayari, I. Modlisation et Commande neuronales dune unit de rgulation de niveau deau. *Sciences et Techniques de lAutomatique STA* **2** (3) 3me trimestre, 2005.

[5] Chatti, A., Ayari, I, Borne, P. and Benrejeb, M. On dynamic non linear system control based on inverse neural networks. In: *International Conference on Machine Intelligence*, ACIDCA-ICMI, Tozeur, novembre, 2005.

[6] Borne, P. and Benrejeb, M. Artificial neural networks presentation. In: *Feature from digital processing to knowledge management: New channel of investigation?* (2sd Part) n 6, October 2006, p. 31.

[7] Borne, P. Artificial neural networks learning. In: *Feature from digital processing to knowledge management: New channel of investigation?* (2sd Part) n 9, October 2006, p. 37.

[8] Borne, P. and Benrejeb, M. Artificial neural networks application to modelling and process control. In: *Feature from digital processing to knowledge management: New channel of investigation?* (2sd Part) n 9, October 2006, p. 55.

[9] Eiben, A. E., Aaarts, E. H. L. and K. M. Van hee. Global Convergence of Genetic Algorithms: A Markov Chain Analysis. In: *Parallel Problem Solving from Nature* (Eds.: H.P. Schwefel and R. Manner), 1991.

[10] Gegout, C. and Rossi, F. Initialisation des Réseaux de Neurones Non Récurrents á coefficients rels par Algorithmes Evolutionnistes. *Journées internationales sur Les Réseaux Neuromimétiques et leurs Applications*, 1994, 416–424.

[11] Gegout, C.*Techniques Evolutionnaires pour l'Apprentissage des Reseaux de Neurones Coefficients Reels*, thse de doctorat, Ecole normale suprieure de Lyon, 1996.

[12] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[13] Withley, D., Starkweather, T. and Bogart, C. Genetic Algorithms and Neural Networks: Optimizing Connections and Connectivity. Parallel Computing, 14, 1990.

[14] Man, K.F., Tang, K.S. and Kwong, S. *Genetic Algorithms, Concepts and Design.* Springer, 1999.

[15] Gruau, F. and Whitley, D. Adding learning to the cellular developmental process: a comparative study. *Evolutionary Computation* **1** (3) (1993) 213–233.

[16] Davis, L. *Handbook of Genetic Algorithms.* New York, Van Nostrand Reinhold, 1991.