15th Triennial World Congress, Barcelona, Spain

EVOLUTIONARY DESIGN OF DYNAMIC NEURAL NETWORKS APPLIED TO SYSTEM IDENTIFICATION

Lavinia Ferariu[1] **and Teodor Marcu**[2]

[1]*"Gh. Asachi" Technical University of Iaşi*
Department of Automatic Control and Industrial Informatics
Blvd. D. Mangeron 53A, RO-6600 Iaşi, Romania
Fax: +40-32-214290, *E-mail*: lferaru@ac.tuiasi.ro

[2]*"Gerhard Mercator" University of Duisburg*
Institute of Control Engineering (AKS)
Bismarckstrasse 81 (BB), D-47048 Duisburg, Germany
Fax: +49-203-379 2928, E-mail: t.marcu@uni-duisburg.de

Abstract: The problem of system identification is addressed by means of general neural networks with locally distributed dynamics. These networks are based on both multilayer perceptron and radial basis function structures. Evolutionary algorithms are suggested to select the optimal neural topologies and parameters. The accuracy of the neural models and the complexity of their architectures are evaluated by considering six objective functions organised on a two-level priority hierarchy. The multiobjective optimisation is solved in the Pareto-sense. Special mechanisms are developed, in order to encourage a rapid improvement of the genetic material. Application to a laboratory three-tank system illustrates the approach. *Copyright © 2002 IFAC*

Keywords: nonlinear system identification, dynamic neural networks, multiobjective optimisation, genetic algorithms, three-tank system.

1. INTRODUCTION

Due to their approximation capabilities, Mapping Artificial Neural Networks (MANNs) have been suggested as a promising alternative to solving problems of nonlinear system identification. In order to approximate dynamic nonlinearities, almost all approaches consider a combination between static MANNs and external dynamic elements. Recently, dynamics was included into the neural networks topologies. The resulted dynamic neural networks are able to cope more efficiently with process dynamics and assure a reduction of input space dimensionality. Details are given in the survey of Isermann, *et al*. (1997) and in the literature cited and discussed therein.

The selection of convenient neural models represents a complex problem for which none method can lead to good results for a large variety of practical situations. It is known that complex neural architectures, described by a large set of parameters, can assure a very good approximation over the considered training data set, but are characterised by improper generalisation capabilities. Also, for a given topology, the optimal set of parameters is difficult to find, because the learning procedure can be trapped into a local optimum point.

Many authors suggest evolutionary methods that allow the automatic design of convenient MANNs. Based on a stochastic search and on mechanisms similar to biological evolution, evolutionary algorithms represent efficient optimisation methods with satisfactory results in nonlinear, constraint and multiobjective optimisations (Bäck, *et al*., 1997). Different combinations with neural network techniques were proposed. Some of them, named collaborative combinations, can assist the selection of proper structures and parameters for MANNs.

The paper suggests an evolutionary method dedicated to select convenient topologies and parameters for a new class of MANNs, namely the Dynamic General Neural Networks (DGNNs). The architectures of DGNNs are obtained by including local dynamic

elements into the topology of static general neural networks.

The paper is organized as follows. Section 2 describes the structure and characteristics of dynamic general neural networks. Details regarding the proposed optimisation algorithm are presented in section 3. The applicability of the approach within the framework of system identification problems is investigated in section 4, with respect to a laboratory three-tank system (Amira, 1993). Finally, several conclusions are delivered in section 5.

2. DYNAMIC GENERAL NEURAL NETWORKS

Depending on the characteristics of the hidden neurons, two broad categories of MANNs (Isermann, *et al*., 1997; Liu and Yao, 1996) can be distinguished: Distributed Neural Networks (DNNs) and Local Neural Networks (LNNs).

The hidden neurons of DNNs have a global response. They extrapolate the region beyond the interval where the training data were acquired. The most common DNN is the multilayer perceptron, characterised by sigmoidal activity functions. When a DNN has to learn a new example, all parameters of the neural network must be re-trained, in order to preserve the already achieved knowledge.

In contrast, LNNs have hidden neurons that produce localised responses. The output of a localised neuron is nonzero if the inputs belong to a small region of the input space. Beyond this region, the response of the neuron is zero. Usually, the Gaussian kernel function is used as activity function. In order to learn a new example, only part of the LNN parameters must be re-trained, because only few neurons have non-zero response at the considered input example.

Even if DNNs and LNNs are computationally equivalent and have both the capability to approximate arbitrary continuous nonlinear functions, interpolation problems can be more efficiently solved with sigmoidal functions and extrapolation problems with localised functions (Liu and Yao, 1996). General neural networks include both sigmoidal and Gaussian nodes, taking advantage from the generalisation capabilities of DNNs and the computational efficiency of LNNs.

The present approach extends the static general neural networks to dynamic general neural networks with locally distributed dynamics. Local synaptic feedback, local activation feedback and local output feedback are implemented by including Auto-Regressive Moving Average (ARMA) filters into the neural topology (Fig. 1). These filters are characterised by the discrete transfer function:

$$G(z^{-1}) = w\frac{1+b_0 z^{-1}}{1+a_0 z^{-1}+a_1 z^{-2}} = \frac{P(z^{-1})}{Q(z^{-1})} \qquad (1)$$

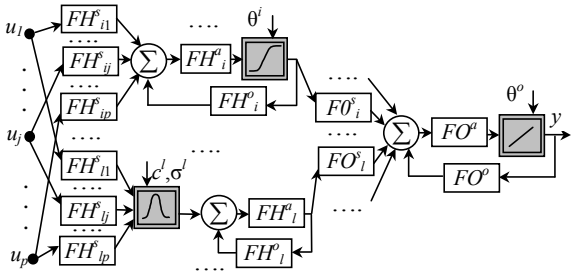

Fig. 1. The topology of DGNN with p inputs, one hidden layer containing n hidden neurons and one output. The hidden neurons are characterised by sigmoidal or Gaussian activity functions. Here FH^s_{ij}, i=1,…,n, j=1,…,p represents the synaptic hidden filter corresponding to the connection considered from input j to hidden neuron i; FH^a_i and FH^o_i denote the activation filter and the output filter of the i^{th} hidden neuron, respectively; FO^s_i specifies the synaptic filter corresponding to the connection considered from the i^{th} hidden neuron to the output neuron; FO^a and FO^o denote the activation filter and the output filter of the output neuron, respectively; θ specifies the bias of the sigmoidal hidden neurons and the bias of the output neuron; c,σ indicate the centres and the standard deviations of the Gaussian hidden neurons.

Here, $P(z^{-1})$, $Q(z^{-1})$ denote the numerator and the denominator, respectively, of the discrete transfer function, and w, b_0, a_1, a_0 represent its parameters. According to equation (1), it is simple to illustrate two particular cases: connection characterised by a simple weight ($a_0 = a_1 = b_0 = 0$, $G(z^{-1}) = w$) and connection eliminated from the neural architecture ($w = 0$). This characteristic allows for a simpler implementation of the proposed design procedure. The topology does not include feedback connections between the neurons included in different layers and lateral connections between the neurons of the same layer. In these conditions, the stability of the dynamic neural networks is easier to check (Isermann, *et al*., 1997).

DGNNs have internal memory and therefore they can cope more efficiently with systems' dynamics. Identification schemes based on DGNNs are built according to the series-parallel model, using input-output process data. They do not include external dynamic elements. That means no a priori knowledge about process dynamic orders and process dead time is required. Also, the dimension of the input space is significantly reduced, because past values of process inputs and outputs are no more necessary to be presented as inputs of the neural model.

The design of convenient DGNNs represents a complex task. First, it is hard to indicate the optimal

structure for a particular case. Second, gradient-based training procedures are difficult to implement, because past states of the network must be considered for gradient computation. The method described in the following uses evolutionary techniques in order to search for the optimal DGNNs. The training procedure does not require the derivatives of the objective functions.

3. EVOLUTIONARY DESIGN OF DGNN

Optimal topologies and parameters of DGNNs are searched for by means of an evolutionary process. At each generation, the algorithm acts on a population of *Nind* possible solutions, named individuals or chromosomes. The initial population is randomly generated from the space of permitted neural models.

Encoding. Each individual included in the population directly encodes the architecture and the parameters of a DGNN. Permitted topologies are described in Fig. 1. All synaptic, activation and output filters are characterised by the discrete transfer function (1). The chromosome is organized in a three–level structure, containing control and parametric genes, as described in Fig. 2. The highest priority level of the considered hierarchy, named level 1, specifies which hidden neurons are included in the encoded neural topology and the type of their activity function (sigmoidal or Gaussian). The second priority level indicates the dynamic structures that are considered in the neural architecture. That level specifies, for each included (active) filter, the orders of the numerator and denominator. All parameters of the DGNN are encoded as real numbers in the lowest priority level, named level 3. The design procedure can consider topologies for which the DGNN input layer is not fully connected with the active hidden neurons, and not all permitted dynamic structures are activated.

The strategy used in a hierarchical genetic algorithm (Mann, *et al*., 1997) is such that the control genes included in a higher level can activate (when control gene's value is nonzero) or deactivate (when control gene's value is „0") the corresponding control or parametric genes contained in a lower level. The inactive genes are preserved in the chromosome structure. These are considered as initial values in the next activations. The hierarchical encoding assures an efficient exploration of the search space, because even small variations of the control genes can produce great changes in the structure of the encoded DGNN. The present approach uses the integer encoding for control levels. By this way, an acceptable compromise is realised between the exploration capabilities of the algorithm and the resulted chromosome length.

Genetic operators are applied independently to each level of genes. Offspring are generated by using discrete recombination (for the control levels 1 and 2), intermediary crossover (for the parametric level 3) and uniform mutation (Bäck, *et al*., 1997). These operators can maintain an appropriate diversity of the genetic material and are able to reduce the negative effect of "competing conventions" (Hancock, 1992). If a resulted offspring does not encode a correct architecture, remedy actions are applied. A correct topology satisfies the following requirements: the hidden layer includes at least one hidden neuron; each hidden neuron is connected to the output neuron and with at least one input of the network; each input is connected to at least one hidden neuron.

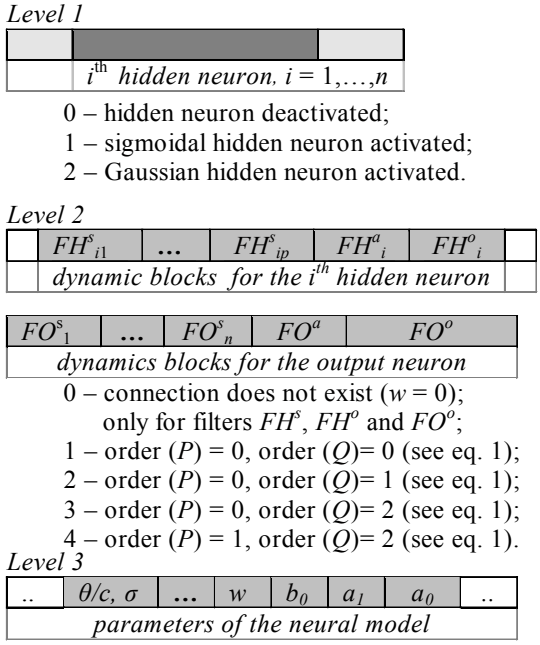

Fig. 2. Hierarchical encoding of the DGNNs. Here $FH^s_{ij}, FH^a_i, FH^o_i, FO^s_i, FO^a, FO^o$ denote the internal dynamic structures illustrated in Fig. 1; w, a_0, a_1, b_0 represent the parameters of discrete transfer functions associated to the internal dynamic elements, as indicated in equation (1); θ specifies the bias of sigmoidal or linear neurons; and c, σ represent the centres and the standard deviations of Gaussian hidden nodes.

Multiobjective optimisation. The design of appropriate neural models is formulated as a problem of multiobjective optimisation. Six objective functions are considered and organized, according to their priority, on a two-level hierarchy. Objective function f_1, namely the sum of output squared errors computed for the whole training data set, represents a measure of the neural model accuracy and is assigned with the highest priority. The values of this objective function are computed after applying a local optimisation procedure, as described later. The other objective functions describe the complexity order of the encoded neural architecture and have the same low-level priority: f_2 - the number of active hidden neurons; f_3 - the number of active connections existing between the network's inputs and the active hidden neurons; f_4 - the number of active output filters; f_5 - the sum of numerators and denominators'

orders, corresponding to all active output filters; f_6 - the sum of numerators and denominators' orders, corresponding to all active synaptic and activation filters.

Pareto-optimisation. The multiobjective optimisation problem is solved in the *Pareto-sense* (Fonseca and Fleming, 1998; Rodriguez-Vazquez and Fleming, 1997). This means that the solution is represented by a family of points, named Pareto-optimal set. Each point of this surface is optimal in the sense that, for all objective functions characterised by the same priority level, no improvement can be achieved in one component without degradation in at least one of the remaining components. The search procedure is combined with a decision mechanism, according to a progressive articulation of preferences (Rodriguez-Vazquez and Fleming, 1997). A goal is associated to each objective function. The goals define the desired area for the objective values. The individuals placed beyond the specified area are not encouraged to produce offspring and to survive. During the evolutionary loop, the goals are adapted according to the mean performances of the current population (Marcu, 1997). The Pareto-optimisation method is based on a ranking selection. The present approach considers new rules for computing the ranks. If an individual satisfies all imposed goals, its rank is assigned based on the values achieved by the objective function f_1. Otherwise, the rank is specified taking into account the degree of goals' violations and the priority of the unsatisfied goals.

The insertion is solved by means of the *Pareto reservation strategy* (Tamaki, *et al.*, 1996). S_{off} offspring and $Nind - S_{off}$ individuals contained in the current population are selected to survive in the next generation. The method encourages the survival of non–dominated chromosomes. If the number of non-dominated individuals exceeds, a parallel selection is considered upon the set of non–dominated individuals, meaning that the selection is applied separately for each objective. If the number of the non-dominated individuals is less than the number of individuals that must survive, the rest of the population is filled in, using a parallel selection applied to the set of dominated chromosomes.

Evaluation of objective function f_1. All DGNNs are trained according to a supervised learning method. Before computing the values of the objective function f_1, a local optimisation procedure is applied, for *NO_ITER* iterations, to the active parametric genes of each chromosome. The local optimisation acts in the Lamarckian-sense and does not require gradient information. As local optimisations, two different methods are used. One method is a standard genetic search and the other one is the optimisation procedure suggested by Salomon (1998). The latter optimisation strategy can be briefly described as follows. At each iteration, the method estimates the gradient direction and the step size, based on the information given by some test candidates.

Migration strategy. In order to assure an efficient search within the large space of permitted neural models, a special mechanism is developed as follows. The population is separated into two partially isolated subpopulations. For the first subpopulation, a mono-objective optimisation is considered, demanding the minimisation of the objective function f_1. The genetic material of the second subpopulation is improved according to the multiobjective optimisation previously formulated. At the end of the evolutionary loop, the best neural model included in the second subpopulation is used. Evolutionary techniques are separately applied to each subpopulation. Once at *NO_MIGR* generations, an appropriate exchange of information is permitted between the considered subpopulations. During migration, the genetic material of the second subpopulation is enriched with individuals well adapted according to the highest priority objective. Also, chromosomes that encode simple neural structures are introduced into the first subpopulation. By this way, the pressure imposed by the highest priority objective is increased and accurate models are encouraged to survive and duplicate.

Evolutionary algorithm of DGNN design. A schematic description is presented as follows:

1. Create an initial hierarchical population containing *Nind* individuals.
2. Check the correctness of the encoded topologies (with remedy actions if necessary) and compute actual values of goals.
3. Apply the local optimisation procedure, for *NO_ITER* iterations, to the active parametric genes of each chromosome.
4. Evaluate the chromosomes according to all considered objectives and compute the fitness values.
5. Loop over a number of *MAX_GEN* generations:
   5.1 For each subpopulation:
      5.1.1 Select parents for the reproduction pool.
      5.1.2 Apply crossover and mutation operators.
      5.1.3 Check the consistency of offspring (with remedy actions if necessary).
      5.1.4 Apply the local optimisation procedure, for *NO_ITER* iterations, to the active parametric genes of each offspring.
      5.1.5 Evaluate offspring and compute their fitness values.
      5.1.6 Insert the offspring into the population, according to the Pareto reservation strategy.
      5.1.7 Once at *NO_MIGR* generations, exchange individuals with the other subpopulation (migration stage).
      5.1.8 Adapt goals and compute fitness values.
   5.2 Determine the best individual(s) of the second subpopulation.
6. Determine best individual(s) over all performed generations.
7. (Optionally) Train with a local optimisation procedure the selected neural model, for a larger number of iterations.
8. End of the algorithm.

4. APPLICATION

The applicability of suggested methodology is studied with respect to the neural identification of the "Three-Tank System DTS 200" (Amira, 1993). This experimental set-up consists of three cylindrical tanks with identical cross sections being filled with water (Fig. 3). The tanks are interconnected with circular pipes. All three tanks are equipped with piezo-resistive pressure transducers for measuring the level of the liquid.

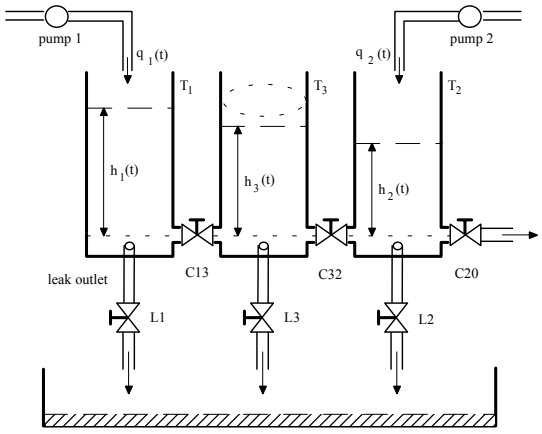

Fig. 3. The "Three-Tank System DTS 200".

The volume flows of lateral tanks, denoted by $q_1(t)$ and $q_2(t)$, represent the two inputs of the system. Three system outputs are considered, namely the liquid levels in the tanks. Two of them, $h_1(t)$ and $h_2(t)$, can be pre-assigned independently. The third output of the process, that is the level $h_3(t)$ in the middle tank, is uncontrollable. Here, t stands for the time variable.

For the experiments, the reference values of the liquid levels were changed pulse-wise. Different magnitudes and periods of rectangular pulses were considered for each controlled tank. The input-output data of the process were sampled at every T_S = 5s during a test period of 400s. The identification task was considered for the normal system behaviour (outlets L_1, L_2, L_3 closed and valves C_{13}, C_{32}, C_{20} opened).

To estimate each process output, a multi-input single output neural network was designed. The DGNN had 5 inputs, representing the current values of process inputs, and the plant output values obtained at the previous sampling moment. In all experiments, a reduced number of hidden neurons was sufficient, i.e. $n = 3$. This allowed for a fast evaluation of the DGNNs. The parameters w, a_0, a_1, b_0, θ were selected between – 5 and 5, and the parameters c, σ between – 1.25 and 1.25. The investigations tested different values for the following parameters: the number of training epochs allowed for the local optimisation procedure (*NO_ITER*), the number of offspring generated at each iteration by the local optimisation procedure (*Noff*), and the number of individuals contained in the population (*Nind*).

It is advantageous to set low values for *NO_ITER*, even in combination with a high population size, because the local optimisation procedure must be applied sequentially, iteration by iteration, to each chromosome, but the genetic search can support a parallel implementation. If the *NO_ITER* value is too low, the topology of the best neural model can result very different from a generation to another one.

The procedure can offer, as final solution, sometimes very simple architectures and sometimes very complex and bad adapted architectures. This „unstable" behaviour is due to the fact that the evaluation is always made on insufficient trained networks. Low values for *Nind* cannot support an efficient exploration of the search space and the obtained results are unsatisfactory.

Table 1. Sum of squared errors (corresponding to normalised training sets) obtained by considering two different local optimisation methods

The estimated plant output	$\hat{h}_1$	$\hat{h}_2$	$\hat{h}_3$
With Salomon's method	0.0328	0.0063	0.0063
With genetic optimisation	0.0356	0.0068	0.0068

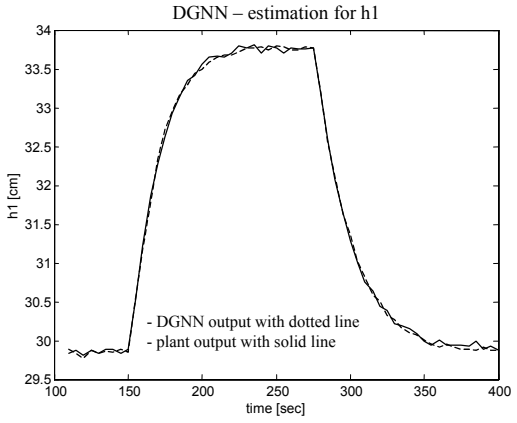

Fig. 4. Model validation (generalisation). The selected neural topology contains one sigmoidal hidden neuron and one Gaussian hidden neuron, and is characterised by the following objective values: $f_2 = 2$, $f_3 = 6$, $f_4 = 1$, $f_5 = 1$, $f_6 = 11$. The active dynamic elements are indicated next: two synaptic hidden filters (FH_{12}^s, FH_{21}^s), all activation hidden filters (FH_1^a, FH_2^a) and all filters corresponding to the output neuron (FO_1^s, FO_2^s, FO^a, FO^o). The sum of output squared errors, computed for the normalised testing data set, is 0.04.

The experiments revealed that an important improvement of the convergence rate can be obtained by using the strategy described in section 3, based on the subpopulations' evolution. Some of the resulted topologies are analysed next. They correspond to the following algorithm parameters: $Nind = 500$, $NO\_ITER = 75$, $Noff = 75$, $NO\_MIGR = 20$, $MAX\_GEN = 300$. The designed neural models are characterised by high accuracy, as indicated in Table 1 and Fig. 4. Their structures are simple, containing few neurons, few connections and few internal dynamic elements. The best results were obtained by using the Salomon's optimisation procedure.

5. CONCLUSIONS

The paper suggests an evolutionary method developed to design convenient general neural networks with locally distributed dynamics. These dynamic networks include two types of hidden neurons, sigmoidal and Gaussian, therefore they can cope efficiently with a large variety of system identification problems.

The design procedure does not require any information about the gradient of the objective functions. Near-optimal neural models are obtained, characterised by a good accuracy and simple architectures. The user must set some parameters of the evolutionary search, but this task seems to be easier than that of manually selecting the DGNNs' topologies. The disadvantage of this procedure is that it needs large computational resources.

Further research will include the design of a fault diagnosis system, based on neural observer schemes (Marcu, *et al*., 1999). In order to guarantee an early detection and isolation of faults, the neural observers must capture the dynamic behaviour of the process, by learning the general trend of the target values and filtering the noise. The global recognition rate of the fault diagnosis system is strongly dependent on the performances of the designed neural observers.

Previous results (Marcu, *et al*., 1999) indicate that, in the case of the "Three–Tank System DTS 200", a fault diagnosis system based on neural observer schemes can detect incipient faults (clogs and leaks) corresponding to mass flows of about 10-20 ml/s. These performances are better than those characterising a fault diagnosis system with linear observers (40 ml/s). Considering the evolutionary method described in the present paper, it is expected to improve the quality of neural observers and, in consequence, the global recognition rate.

Acknowledgements. The first author gratefully acknowledges the support of this research by the project "Laboratory for Training in Computer-Aided Engineering" (LICAP-2 MU 103), financed by the World Bank and the Romanian Government. The second author would like to acknowledge the support for the completion of this work by the EU-IST-2000-30009 project "Multi-Agents-Based Diagnostic Data Acquisition and Management in Complex Systems" (MAGIC).

REFERENCES

Amira (1993). *Laboratory Experiment Three-Tank System*. Amira GmbH, Duisburg, Germany.

Bäck, T., D. Fogel and Z. Michalewicz (1997). *Handbook of Evolutionary Computation*. Oxford University Press, UK.

Fonseca, C.M. and P.J. Fleming (1998). Multiobjective Optimisation and Multiple Constraint Handling with Evolutionary Algorithms – Part I: A Unified Formulation. *IEEE Transactions on Systems, Man, and Cybernetics – Part A*, **28** (1), 26-37.

Hancock, P.J.B. (1992). Genetic Algorithms and Permutation Problems: A Comparison of Recombination Operators for Neural Net Structure Specification. Proceedings of *IEEE International Workshop on Combinations of Genetic Algorithms and Neural Networks*, Baltimore, Maryland, pp.108-122.

Isermann, R, S. Ernst and O. Nelles (1997). Identification with Dynamic Neural Networks: Architectures, Comparisons, Applications. Preprints of *IFAC Symposium on System Identification,* Fukuoka, Japan, Vol.3, pp. 997-1022.

Liu, Y. and X. Yao. (1996). Evolutionary Design of Artificial Neural Networks with Different Nodes. Proceedings of *Conference on Evolutionary Computation*, Nagoya, Japan, pp. 670-675.

Mann, K.F., K.S. Tang, S. Kwong and W.A. Halang (1997). *Genetic Algorithms for Control and Signal Processing*. Springer, London, UK.

Marcu, T. (1997). A Multiobjective Evolutionary Approach to Pattern Recognition for Robust Diagnosis of Process Faults. Preprints of *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Hull, UK, Vol.2, pp.1193-1198.

Marcu, T., L. Ferariu and P.M. Frank (1999). Genetic Evolving of Dynamic Neural Networks with Application to Process Fault Diagnosis. Proceedings of *European Control Conference*, Karlsruhe, Germany, CD-ROM, F1046-1.

Rodriguez-Vazquez, K. and P.J. Fleming (1997). A Genetic Programming NARMAX Approach to Nonlinear System Identification. Proceedings of *GALESIA'97*, Sheffield, UK, pp.409-413.

Salomon, R. (1998). Evolutionary Algorithms and Gradient Search: Similarities and Differencies. *IEEE Transactions on Evolutionary Computation*, **2** (2), 45-55.

Tamaki, H., H. Kita and S. Kobayashi (1996). Multiobjective Optimisation by Genetic Algorithms: A Review. Proceedings of *Conference on Evolutionary Computation*, Nagoya, Japan, pp. 517-522.