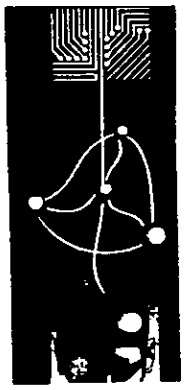


1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajuba. 24 a 27 de outubro de 1994

Autonomous Control Using Soft Computing Paradigms

M. De Oliveira **M. Figueiredo** **F. Gomide**
UNICAMP/FEE/DCA
Campinas, São Paulo, Brasil

L. Romero
Universidad de Valladolid/Faculdad de Ciencias
47011 - Valladolid, Spain

assfalk@fee.unicamp.br mauri@dca.fee.unicamp.br
gomide@dca.fee.unicamp.br
luis@venus.isa.cie.uva.es

Abstract

In this paper, a genetic algorithm has been used to design of neurofuzzy systems as an alternative to traditional nonsupervised learning methods for neural networks. Fuzzy systems and neural networks also get improved behaviour with their symbioses. Here, we have integrated neural networks, fuzzy systems and evolutionary computing techniques to developed a control system for an automatically guided vehicle (AGV). The network learns to guide the AGV to given goals without collisions. The knowledge learned by the network can be extracted as if-then fuzzy rules and the number of rules, number of partitions and the shape and position of membership functions are easily determined.

1. Introduction

Recently, researchers have studied the symbiosis of genetic algorithm techniques, fuzzy set theory and neural networks. For instance, genetic algorithms have been used to design fuzzy systems.This determines membership functions, consequent parts and the number of *if-then* fuzzy rules [Tak93a]. Likewise fuzzy set theory is being used to adjust some parameters of genetic algorithm such as population size, mutation rate and crossover rate [Tak93b].

Another interesting integration concerns fuzzy set theory and neural networks [Lin91]. The resulting system is a neurofuzzy network, which shares the benefits of the original systems: parallel computation, robustness, natural language processing and imprecise data processing [Fig93].

Genetic algorithms have been considered as an alternative learning method for neural networks, filling the nonsupervised learning method gap [Ich92, Mac92].

In this paper we present a controller for an automatically guided vehicle (AGV)

which integrates the above three areas, that is, by using techniques of soft computing. The network learns to guide the AGV to goals without colliding with environment obstacles. The knowledge acquired by the network can be extracted as *if-then* fuzzy rules since it also determines the number of rules, membership functions and consequent parts. This neurofuzzy model is an attractive and convenient approach for designing fuzzy systems.

2. A Neurofuzzy Network

The fuzzy neural network used here will be briefly described below. For more details please refer to [Fig93], where it first appeared. The structure of the system under discussion will be centered around a set of "if-then" conditional statements (rules) given as follows:

input premise: X_1 is A_1 and X_M is A_M

rule 1: if X_1 is A_1^1 and X_M is A_M^1 then y is g^1

rule M: if X_1 is A_1^N and X_M is A_M^N then y is g^N

consequence: y is g

where: X_j is a fuzzy variable, A_j and A_j^i are the corresponding fuzzy sets, while "y" is a real variable, g^i are viewed as constants defined in the output space.

All the universes are assumed to be discrete. To simplify the notation we make z_k the grade of membership of Z at $x_k(Z(x_k) = z_k)$ with x_k denoting a numerical value in the input space.

The control decision y is determined via a sequence of the three reasoning stages: matching, antecedent aggregation and rule aggregation.

These steps are carried out by specialized neurons, the min-max neurons [Gom92], which model more closely their biological counterparts, by incorporating more complex decodification function, synaptical and input aggregation operators [Roc92].

The proposed fuzzy neural network (see Fig. 2.1) has a feedforward architecture with five layers of neurons. The first layer implements input fuzzification, the second rule antecedent matching, the third antecedent aggregation, the fourth consequent aggregation. The last layer supplies the final output, as a weighted average of each rule output contribution.

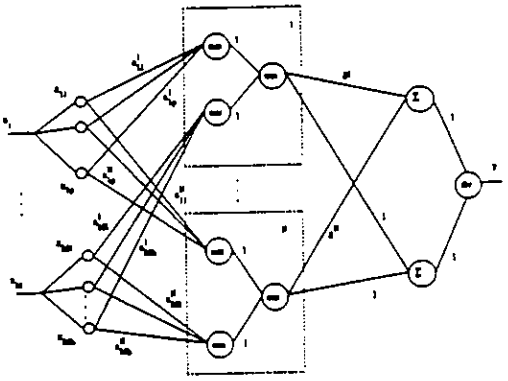


Figure 2.1: The fuzzy neural network.

3.A Brief Overview of GA

A genetical algorithm (GA) is basically a search procedure based on biological evolutionary mechanisms: natural selection, genetic recombination and mutation.

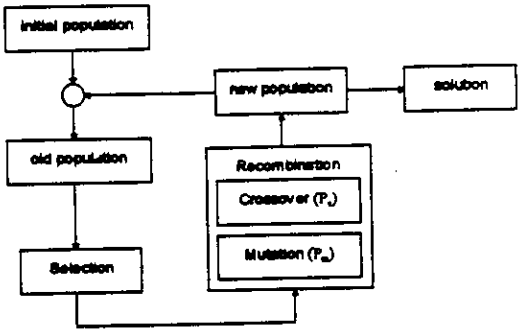


Figure 3.1 The basic Genetic Algorithm.

As in its natural counterpart, the GA operates upon a population of candidate solutions, a sample of points in the solution-space. Each individual is represented by its operational characteristics, in a coded form, the *chromosome*. This chromosome may be thought of as the coordinates of the

individuals in the solution space. The chromosome may be a string of bits (most common), of real or integer numbers or even an array of numbers (as in [Ich92]).

The search procedure is outlined in Figure 3.1. At first, the whole population is evaluated singly and each individual is awarded a *fitness* function, based upon its performance as a solution. Then, a set of individuals (usually two) are selected, according to their fitness function, for the re-combination operation. Recombination consists of two operations: *crossover* and *mutation*. These operations occur with probabilities P_c and P_m, respectively.

Crossover promotes the exchange of parts of the involved individuals' chromosome. One or more crossover sites are selected and the genetic substrings thus defined are switched among the individuals (see Fig. 3.2). The principle (the *building blocks* principle) behind this exchange of genetic material is that high fitness individuals owe their good performance to good characteristics coded in their chromosomes. Thus the recombination of good building blocks would result in better individuals. The mutation operator adds a variable amount of noise to the chromosome, generating new varieties of solutions.

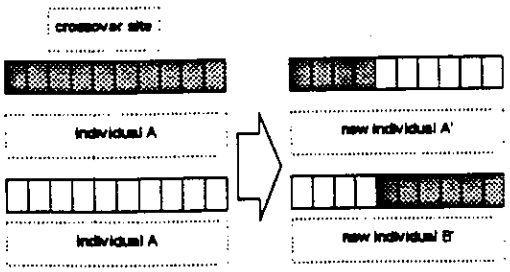


Figure 3.2 Single Site Crossover.

The selection and recombination process is repeated until a given amount of new candidate solutions are generated. This percentage, the *generation gap (G)*, is usually defined in terms of a fraction of the size of the original population. A new population is formed by (1-G)*sizeof(old population) individuals of the old population plus the newly-generated individuals. A variation of this scheme involves keeping the best individual of the old population. This is known as *elitism*, and usually is used with high gap ($G \rightarrow 1$) variations of the GA.

Once a new population is formed, the process is repeated, until a satisfactory solution is created or until a given number of population iterations, known as *generations*, is executed.

It can be shown that the population maintained by the GA converges to an optimal solution [Qi94].

4.Neurofuzzy learning employing a GA

The design of neural networks using GA techniques has been considered by many authors. At present, three main research fields may be identified: the determination of the overall structure of the neural network (number of layers and neurons, as well as the connections between them), the optimization of traditional training algorithm parameters and the optimization of the synaptic weights.

Ichikawa and Sawa's use of GA [Ich92] for training a neural-network is an example of the synaptic-type GA-neural network integration. They use an integer array-codified neural network as a population member. The characteristics encoded were the synaptic weights, while the structure remained the same throughout each case problem. Since their neural network was inserted in a feedback control loop, they could not rely upon the usual (input, output) comparison fitness function. Instead they built a

fitness function that evaluated the behaviour of the controlled plant during the simulation, against that what would be desired.

Harp et al. [Har89]took the overall structure track and developed a GA software package that tuned both the backpropagation parameters and the structural characteristics of the neural netowrk. Once again, the feed-forward network model was used.

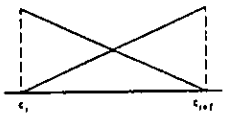
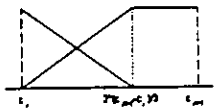
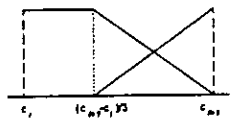
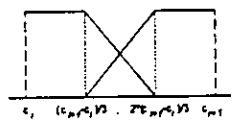
In our approach both the synaptic-optimization and structure determination are applied to the neurofuzzy network.

The GA employed is an elitist model operating upon an integer/real chromosome. The characteristics tuned by the GA over the generations are the rule base consequents and the membership functions, given by shape and position. The rules consequents are represented as real genes, while the membership function representation is more complex, contained in a 2-tuple (center, type), where center determines the whereabout of the function centre point and type is either trapezoid shape or triangular shape (see Table 3.1). Thus, for an n-rule rule base, k antecedents and f_i membership functions per antecedent, the chromosome describing this neurofuzzy network has $n+\sum_{i=1}^{k} f_i$ real genes and $\sum_{i=1}^{k} f_i$ integer genes.

The initial population is generated randomly. The membership function genes are interpreted according to its shape and center point and its immediate neighbours' shapes and center points. The previous function determines the left side of the current function and the following function the right side of the current function. See Table 3.1.

Crossover always occurs, selecting only one site. The individual selection uses the weighted roulette algorithm, which confers higher selection probabilities to individuals with higher fitness. The mutation operator is quiescent for most of the run (probability zero).

Table 3.1 Membership Function Shaping

Function (i)'s shape (center point is c_i)	Function $(i+1)$'s shape (center point is c_{i+1})	Function (i)'s right side and Function $(i+1)$'s left side
Triangular	Triangular	
Triangular	Trapezoid	
Trapezoid	Triangular	
Trapezoid	Trapezoid	

The mutation probability jumps abruptly to 0.1, for one generation, when the average population fitness becomes greater than or equal to 90% of the best population fitness. This population "shake-up" creates individuals with new genetic traits, staving off population staleness. Our convergence-triggered mutation operator is based on Kauffman's strategy of periodic mutational bursting [Kau86].

Each individual was evaluated by placing it in a simulated environment containing obstacles and targets for a given amount of time or until it collided

with an obstacle. The objective function is given as a function of the number of targets reached, the mean distance covered and the distance to the target at the end of a simulation. The fitness function value is generated by scaling the population objective function values.

5. Results

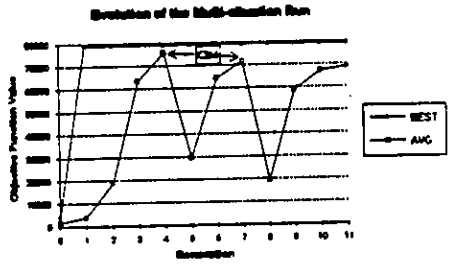


Figure 5.1 Objective Function Evolution,Run A.

Two variants of the GA described in section 4 were used. Variant A had a 104 member population and used a simulation with 3500 time steps and presenting 4 obstacles in 4 sequences ((A,B,C,D), (B,C,D,A) etc.). The objective function was taken as an average of the objective functions over each sequence.

Variant B had a 64 member population and used a simulation with 2000 time steps, presenting only one sequence with four obstacles (A,B,C,D). The convergence behaviour can be seen in Fig.5.1.

Variant A presented the best results, given its more thorough and varied objective function evaluation (Fig.5.2). The AGV was able to navigate successfully even when the targets were placed in different places (Fig.5.3).

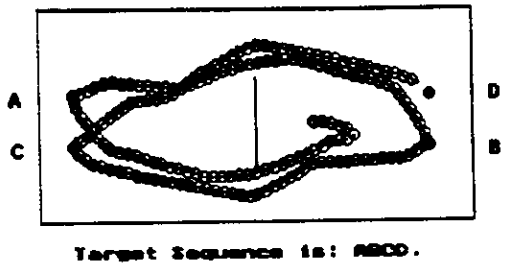


Figure 5.2 Best individual run, Variant A.

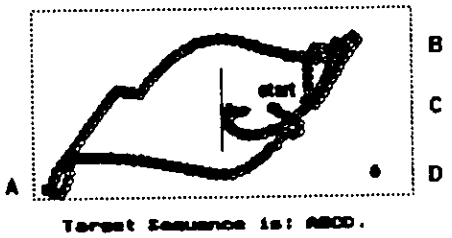


Figure 5.3 Different targets, Variant A

6. Conclusion

We introduced in this paper a type of genetic algorithm as an alternative learning method for neurofuzzy networks. The method was evaluated using the AGV autonomous control problem. Simulation results show that the network successfully guides the vehicle to the goals without colliding.

We have also investigated the control of mutation rate using the degree of population convergence. Mutation occurs only if the population convergence is greater than a given threshold value.

After the learning period we were able to extract the knowledge of the network as if-then fuzzy rules (number of rules, partitions and membership functions).

Currently we are seeking to progressively increase the complexity of the problem, using two approaches:

- incorporating a set of several, increasingly complex environments in the evaluation;

• implementing several runs, each with increasingly difficult lay-outs, where a part of the initial population of each run would be taken from a sample of the preceding run.

Acknowledgements

The authors would like to acknowledge CAPES, FAPESP 93/3034-8, CNPq #300729/86-3 and ESPRIT-ECLA005 for their support.

References

[Fig93] M. Figueiredo, F. Gomide and W. Pedrycz, "A Fuzzy Neural Network: Structure and Learning", 5th IFSA World Congress, Seul, Korea, 1993.

[Gom92] F. Gomide and A. Rocha, "A Neurofuzzy Components Based on Threshold", IFAC, Silica, Spain, 1992.

[Gol89] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, 1989.

[Har89] S. Harp, T. Samad and A. Guha, "Toward Genetical Synthesis of Neural Networks", *Proceedings of ICGA'89*, pp.360-369.

[Ich92] Y. Ichikawa and T. Sawa "Neural Network Application for Direct Feedback Controllers", IEEE Transactions on Neural Networks, Vol.3. No.2, March 1992.

[Kau86] S.A. Kauffman and R. Smith "Adaptive Automata Based on Darwinian Selection", Physica 22D, pp. 68-82, 1986.

[Lin91] C.-T. Lin and C.S.G. Lee,"Neural-Network-Based Fuzzy Logic Control and Decision System", IEEE Transactions on Computers, vol.40, no.12, December, 1991.

[Mac92] R. Machado and A. Rocha, "Evolutive Fuzzy Networks", Proc. of the International Conference on Fuzzy Systems, pp.493-500, San Diego, USA, 1992.

[Qi94] X. Qi and F. Palmieri, "Theoretical Analysis of Evolutionary Algorithms With a Infinite Population Size in Continuous Space, Parts I & II", IEEE Transactions on Neural Networks, Vol.5, No.1, pp.102-129, Jan.1994.

[Roc92] A.F. Rocha, Neural Nets: a theory for brains and machines, Springer Verlag, 1992.

[Tak93a] H.Takagi, "Neural Networks and Genetic Algorithm Approaches to auto-design of Fuzzy Sytems", Proc. of Fuzzy Logic in Artificial Intelligence, Linz, Austria, June, 1993.

[Tak93b] H. Takagi, "Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques", Proceedings of the 5th Int'l Conf. on Genetic Algorithms, Urbana-Champaign, IL, July,1993.