Robot Arm Fuzzy Control by a Neuro-Genetic Algorithm

Carlos Kavka, María Liz Crespo
Proyecto UNSL 338403 *
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950, 5700, San Luis, Argentina
E-mail: {ckavka,mcrespo}@unsl.edu.ar

Abstract

Robot arm control is a difficult problem. Fuzzy controllers have been applied succesfully to this control task. However, the definition of the rule base and the membership functions is itself a big problem.

In this paper, an extension of a previously proposed algorithm based on neuro-genetic techniques is introduced and evaluated in a robot arm control problem.

The extended algorithm can be used to generate a complete fuzzy rule base from scratch, and to define the number and shape of the membership functions of the output variables. However, in most control tasks, there are some rules and some membership functions that are obvious and can be defined manually. The algorithm can be used to extend this minimal set of fuzzy rules and membership functions, by adding new rules and new membership functions as needed.

A neural network based algorithm can then be used to enhance the quality of the fuzzy controllers, by fine tuning the membership functions.

The approach was evaluated in control tasks by using a robot emulator of a Philips Puma like robot called OSCAR. The fuzzy controllers generated showed to be very effective to control the arm.

A complete graphical development system, together with the emulator and examples is available in Internet.

Keywords: robot arm control, fuzzy controllers, neural networks, evolutionary algorithms.

*The research group is supported by the UNSL and the ANPCYT (Agencia Nacional para la promoción de la Ciencia y Técnica

1 Robot emulator

The robot emulator selected was Simderella, developed by Patrick van der Smagt at the University of Amsterdam in the Netherlands [11]. This system emulates a Philips PUMA like robot called OSCAR with three rotational joints. A diagram is presented in figure 1. The joint 3 is coupled with the joint 2, providing the effect that when the joint 2 is moved, the joint 3 is moved automatically in the opposite direction, in such a way that the angle of the third link of the arm and the base remain constant.

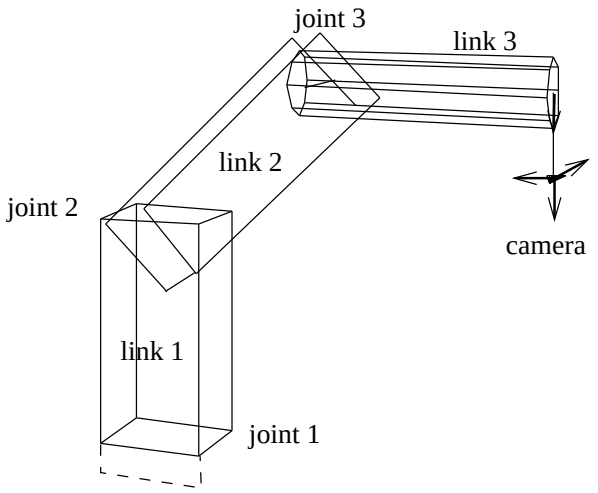

Figure 1: A diagram of OSCAR

The robot arm has a camera located in the end effector (hand) that provides the distances to the target object in all x, y and z coordinates. Distances can be positive or negative depending on the position of the hand relative to the target object.

The emulator provides commands to reset the arm to a default position, to move the arm by providing the offsets to apply to the joints, and to read the distances to the target object.

2 Control Problem

The control task can be specified as follows: given a random starting position for the arm and a random position for the target object, move the arm to reach the target object.

The camera located in the hand of the robot provides the distance to the target object at any time. The goal state is such that the distance in the x direction is 0, the distance in the y direction is 0 and the distance in the z direction is 0.

The objective is to build a controller that can provide the offsets to apply successively to the joints 1 and 2 to move the arm to reach the target object. The controller should have three inputs and two outputs. The three inputs are the distance in the x direction to the target object, the distance in the y direction and the distance in the z direction. The outputs are the offsets to apply to the joints 1 and 2. A diagram is presented in figure 2.

Figure 2: Controller

The space in which the robot operates is specified as a standard three dimensional space with three coordinates corresponding to the axis x, y and z. The joint 1 is always located at the origin. The robot arm cannot go below the floor so z values are always positive. We restrict the x values to be positive. The resulting range for x is 0..100, for y is -100..100 and for z 0..100.

The target object is located in random positions in the space, but avoiding positions in which the arm cannot reach it. The distances obtained by the camera can be in the range -100..100 for x, -200..200 for y and -100..100 for z. Figure 3 presents some examples.

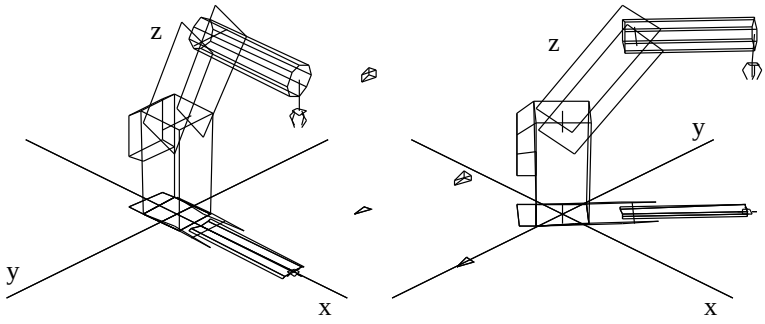

Figure 3: Some examples: The robot arms shown here have the hands in (70,0,70) and (55,60,65) respectively, the target objects are in (55,60,65) and (0,-60,40) respectively and the distances from the hands to the target objects are (15,60,5) and (125.64,-40.54,25) respectively.

The outputs of the control system are the offset to be applied to the joints. They can be positive or negative, and we restrict their values as float values in the range -1..1, to allow smooth movements of the arm.

3 Fuzzy Control

The first step in the definition of a fuzzy controller is the selection of the fuzzy membership functions for the input variables. We propose to chose a standard partition with the three fuzzy sets *negative*, *zero* and positive for each input variable, as it is show in figure 4.

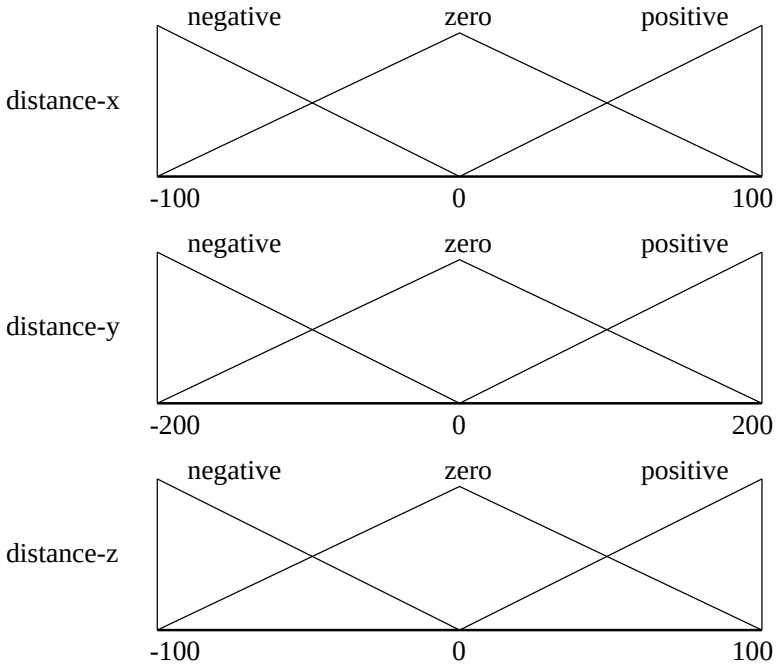

Figure 4: Fuzzy sets for the three input variables

The selection of the fuzzy membership functions for the output variables and the rule base construction is a more difficult problem. Even if we select a standard partition of seven fuzzy sets for each output variable, the definition of the rules to consider which action should be taken for both output variables in all possible combination of inputs is quite difficult.

The algorithm that we propose here, allows to generate automatically the output fuzzy sets and the rule base.

4 Symbiotic Evolution

In a previous work, we have proposed to use symbiotic evolution to evolve neural networks that represents fuzzy systems [13]. The advantage to evolve neural networks is that at the end, we obtain a fuzzy controller that can be enhanced by traditional neural network methods.

Traditional developments in genetic algorithms consider each individual in the population as a complete neural network, so all of them are evalu-

ated on an independent basis. The population converges to the dominant structure. This convergence is desirable if it goes into the global optimum, however, usually it converges into a local optimum.

When the population has converged prematurely, the genetic algorithm does a linear search in the solution space by just doing mutations, instead of doing an efficient parallel search.

The basic idea of the symbiotic evolution is that the individuals of the population will represent partial solutions. The goal of each individual is to be an adequate partial solution to be combined with other individuals of the population. As each individual is not a solution by itself, and as it must rely on others to reach a high fitness value, it must maintain a symbiotic relation.

Each partial solution must be specialized in one aspect of the problem, so in this way, it is impossible to have a premature convergence of the population. The population remains diverse.

The SANE (Symbiotic Adaptive Neuro Evolution) algorithm was proposed by Miikkulainen and Moriarty [1] [2]. The novel approach of SANE, is that it encodes one unit of a neural network as one string (chromosome). The fitness of a unit is determined by its degree of cooperation with the other units used to form the network. SANE keeps a population of units that represents the hidden units in a standard feed-forward neural network. The input and output units are determined by the problem itself.

SANE is very good in doing a quick effective search of the solution space, but it becomes very difficult for it to reach the best solution.

HSANE (Hierarchical SANE) is proposed by the same authors and overcomes this difficulty. It combines the advantages of the network level evolution with the advantages of the unit level evolution. HSANE keeps two different populations, one for the units, and another one for *network blueprints*. The population of units evolve into good units, and the population of networks into good combinations of units.

5 Representation of fuzzy systems as neural networks

A fuzzy system can be represented as a five layers GARIC-like neural network [5]. For example, a simple fuzzy system with two input variables with three fuzzy sets each one, two output variables with two fuzzy sets each one, and five rules, can be represented as it is shown in figure 5.

The first layer contains one unit for each input variable. The second layer contains one unit for each fuzzy set of the input variables. The third layer one unit for each fuzzy rule. The fourth layer one unit for each fuzzy set of the output variable. The fifth just one output unit. The connection model between layers two, three and four is based on the fuzzy rules set.

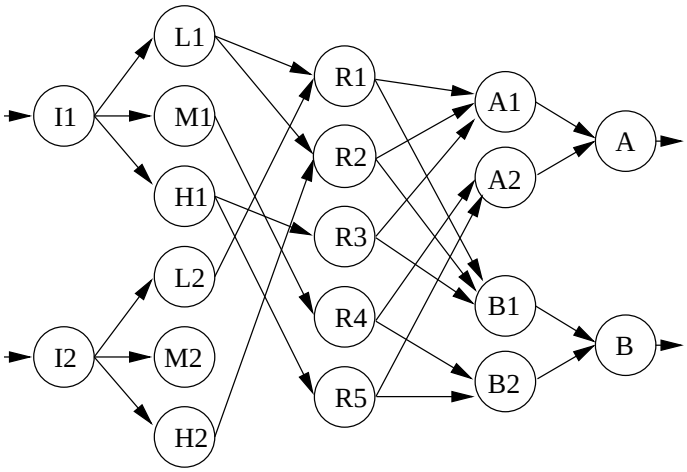

Figure 5: Fuzzy system as a GARIC-like neural network

Just as an example, the rule 2 represented by the neural network in the figure 5 is:

```
if I1 is L1 and I2 is H2 then A is A1 and B is B1.
```

6 Codification

A new codification should be defined to apply HSANE to the domain of the fuzzy neural networks that represents fuzzy systems with more than one output variable. Note that with a symbiotic approach, one string will not represent a complete neural network.

The layer one, two and five are not included in the codification because they are fixed. Only the layers three and four, and the connection pattern will be codified into the strings. One string will represent one unit for each output variable of the fourth layer with all the units in the layer three to which they are connected, and all the connections between them.

The following parameters are defined:

- N: this is the number of chromosomes used to form a network and it will determine the size of the fuzzy system obtained.
- OV: this is the number of output variables and it is fixed depending on the problem. In the case of the robot control it will be 2.
- IV: This is the number of input variables and it is also fixed depending on the problem. In the case of the robot control it will be 3.
- NR: This is the maximum number of rules that can use one particular fuzzy set of an output variable. The evolutionary process will consider

this limit and it will not generate more rules with the same consequent. This is particularly important when the fuzzy system will be implemented in hardware, where this restrictions apply.

The figure 6 shows two strings that codify the network from figure 5.

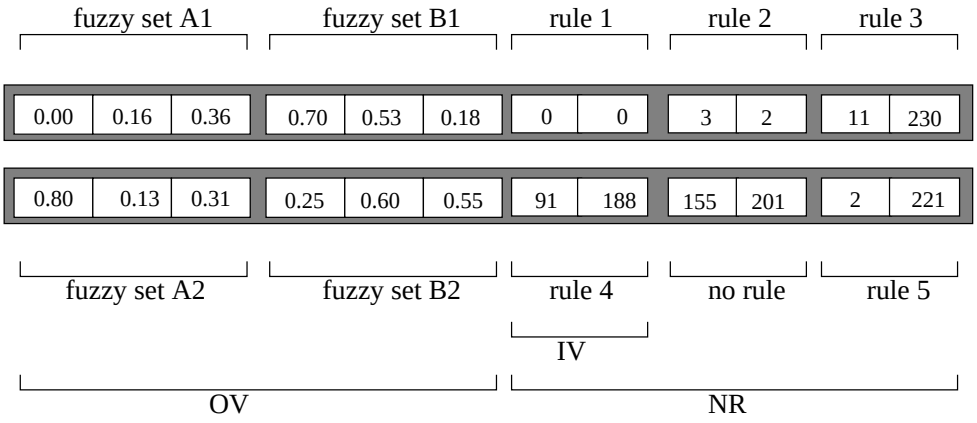

Figure 6: Strings that codify the network from figure 5

A string consists of a series of fields. It starts with OV groups of three fields. These three first fields (16 bits values) will codify three floating point numbers in the range 0...1, that represent respectively the center, left spread and right spread of the fuzzy set defined by a fourth layer unit (output fuzzy set). In this way, one output fuzzy set for each output variable will be codified at the beginning of the string.

Following them, a series of NR groups of IV 8 bits fields will codify how the connections must be done to the units in layer three. Let k_i $(i = 1, ..., n)$ be the number of fuzzy sets for the input variable i. If the value d of a field is smaller than or equal to 127, then $d\ modulo(k_i)$ is the index of the input fuzzy set of the variable i. This means that this particular fuzzy set is an antecedent of the rule node under consideration. If the value d is bigger than 127, then no connection is made to this input variable. It is assumed that the input fuzzy sets for each variable are numbered starting from zero.

The output fuzzy sets codified by these strings are shown in figure 7. The areas of the fuzzy sets that are outside the range 0...1 are not considered at all.

7 Learning strategy

The evolutionary algorithm starts by creating the two populations, and doing the corresponding crossover and mutation operations. Each individual in the networks population is evaluated by creating the fuzzy system it represents, and applying it to the control problem.

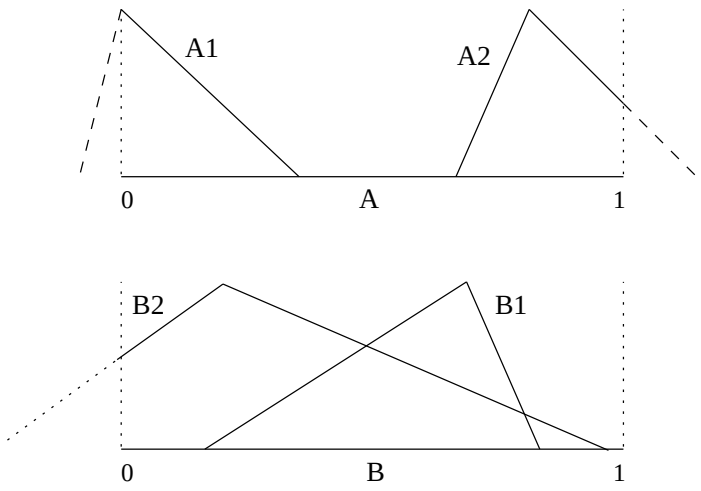

Figure 7: Output fuzzy sets codified by the strings from figure 6

Each system is evaluated in 100 time steps, always starting from the rest position with the hand in (85,0,65). Four different test cases are considered with the target object in (70,60,55), (55,45,75), (70,-60,55) and (55,-45,75). Only four cases are enough, because they include all possible movements for the arm. They include cases to move the arm in the right-down direction, in the left-down direction, in the left-up and in the right-up direction.

The fuzzy controller obtained should be able to reach a target object located in any position starting from any position, even if it was generated by just considering these four cases. The reason is that the controller does not know anything at all about absolute coordinates. It only considers distances and offsets.

An incremental learning strategy is proposed to obtain controllers that can to reach the target object as fast as possible. The arm is allowed to be at least at 50% of the maximum distance at 50 time steps and at least at 25% of the maximum distance from the moment in which the 75 time steps are reached. If it fails to fulfill these requirements the evaluation process for the current test case is abandoned.

The fitness function is defined in such a way that controllers that can reach the target object receives higher score the controllers that fail to reach it.

$$\text{fitness(system)} = \sum_{k=1}^{4} (\text{maxX} - x_t)^2 * (\text{maxY} - y_t)^2 * (\text{maxZ} - z_t)^2$$

where k is the evaluation index, $maxX$ the maximum value allowed for the distance in the x axis, and x is the distance provided by the camera at the end of the movement sequence. $maxY$, $maxZ$, y and z are defined in an analogous way.

parameter	value
low level population size	800
high level population size	100
fuzzy sets per output variable	7
maximum number of rules per fuzzy set	2
low level individuals for crossover	150
high level individuals for crossover	30
mutation rate	0.1

Table 1: Parameters for the evolutionary algorithm

The parameters of the evolutionary algorithm are defined as it is shown in table 1.

8 Experimental results

All runs of the evolutionary process were successful, it means that in every execution of the evolutionary process, a successful controller is obtained. The results presented here corresponds to one example of these runs.

Figure 8 shows the evolution of the fitness against the number of generations.

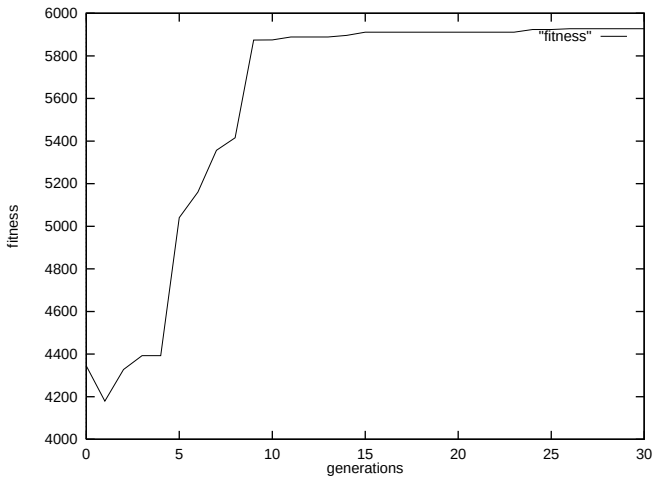

Figure 8: Fitness evolution

Figure 9 to figure 12 shows the plot of the distances when the fuzzy controller obtained in generation 15 is evaluated on the four cases used for learning. The performance is adequate as expected.

The algorithm produces 7 fuzzy sets for each output variable. Their details are presented in table 2.

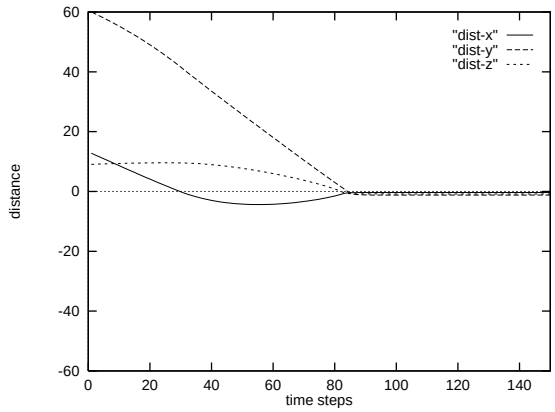

Figure 9: Performance at test case 1

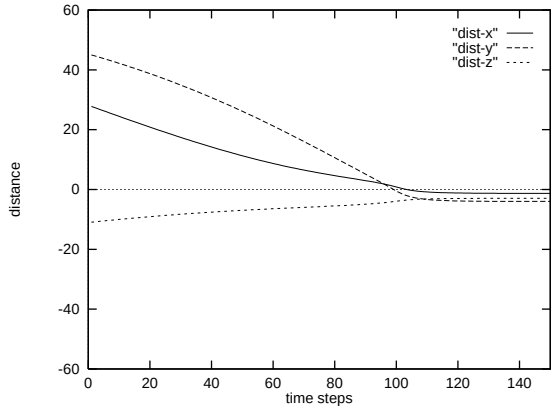

Figure 10: Performance at test case 2

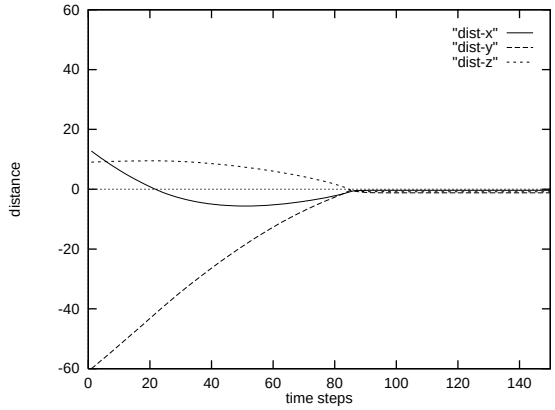

Figure 11: Performance at test case 3

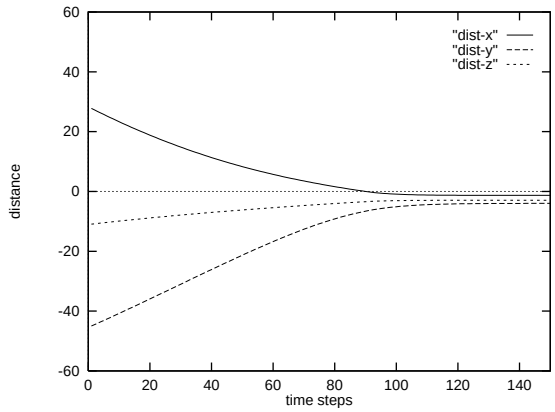

Figure 12: Performance at test case 4

variable	set	center	left spread	right spread
Joint1	Set0	-0.622	1.363	1.663
	Set1	-0.769	0.689	0.126
	Set2	0.344	0.706	1.721
	Set3	0.530	0.907	1.003
	Set4	0.500	0.297	1.547
	Set5	-0.574	1.604	0.292
	Set6	0.647	1.046	1.400
Joint2	Set0	0.618	0.130	0.692
	Set1	-0.463	1.097	1.553
	Set2	0.767	0.414	1.613
	Set3	-0.652	1.392	1.434
	Set4	-0.025	1.079	1.248
	Set5	-0.475	0.882	0.744
	Set6	0.286	1.843	0.958

Table 2: Fuzzy sets produced by the algorithm

The algorithm produced 14 rules. They are presented in table 3. The rules may seem strange, but this is the usual aspect of rules when they are generated through an automatic procedure.

The generated fuzzy controller has also a good performance starting from different random positions, and reaching objects located in different random positions. Some examples are shown in figure 13 to figure 15.

9 The FC-DT system

The algorithm introduced in this paper was included in the FC-DT (Fuzzy Controllers Design Tool) developed by the same authors. FC-DT is a graph-

If dist-y is positive then Joint1 is Set6 and Joint2 is Set6.
If dist-x is positive and dist-z is negative then Joint1 is Set6 and Joint2 is Set6.
If dist-y is negative then Joint1 is Set5 and Joint2 is Set5.
If dist-x is positive then Joint1 is Set5 and Joint2 is Set5.
If dist-z is positive then Joint1 is Set4 and Joint2 is Set4.
If dist-y is positive then Joint1 is Set4 and Joint2 is Set4.
If dist-z is negative then Joint1 is Set3 and Joint2 is Set3.
If dist-z is negative then Joint1 is Set3 and Joint2 is Set3.
If dist-y is positive and dist-z is negative then Joint1 is Set2 and Joint2 is Set2.
If dist-x is negative and dist-y is positive then Joint1 is Set2 and Joint2 is Set2.
If dist-x is zero and dist-y is negative then Joint1 is Set1 and Joint2 is Set1.
If dist-y is positive and dist-z is positive then Joint1 is Set1 and Joint2 is Set2.
If dist-y is negative then Joint1 is Set0 and Joint2 is Set0.
if dist-x is negative then Joint1 is Set0 and Joint2 is Set0.

Table 3: Fuzzy rules produced by the algorithm

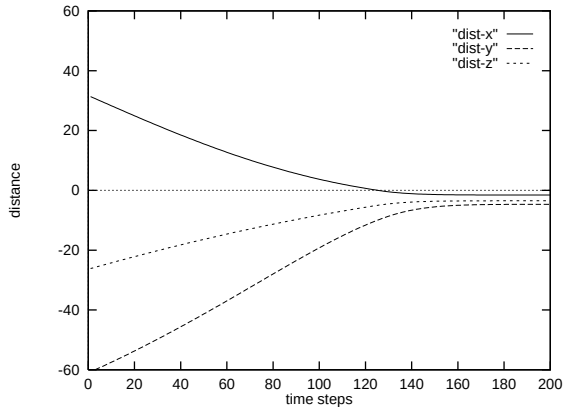

Figure 13: Performance at new case with starting position at (95,25,35) and target position at (80,-45,60)

ical tool that allows the definition and graphical edition of complete fuzzy controllers. Includes the possibility to enhance fuzzy controllers by neural network algorithms and now, to generate rules and fuzzy membership functions for systems with more than one output variable. The fuzzy controllers can also be enhanced by manual edition.

By using the environment introduced by FC-DT, some rules and some membership functions for the output variables can be defined before. For example, it is very easy to imagine that a kind of *zero* fuzzy membership functions will be useful for any controller. Some rules for making no movements when the distances are *zero* could be convenient also. The evolutionary process will create a new controller but always including the rules and

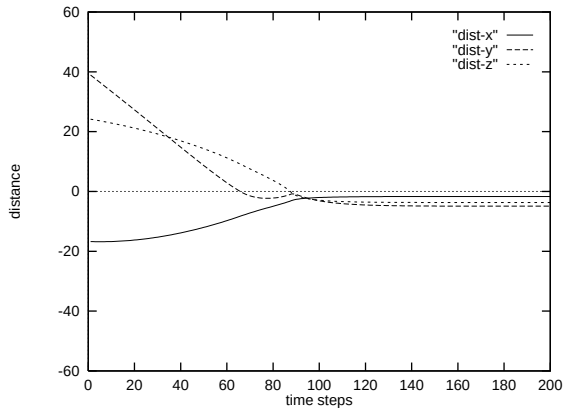

Figure 14: Performance at new case with starting position at (73,-30,75) and target position at (100,5,45)

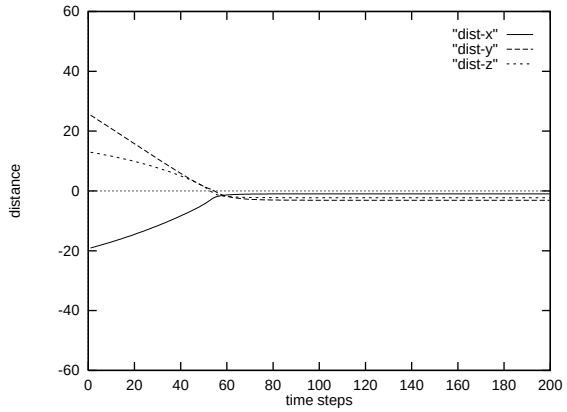

Figure 15: Performance at new case with starting position at (45,-55,75) and target position at (75,-55,60)

fuzzy sets defined previously. This inclusion of obvious rules and obvious membership functions saves a lot of learning time.

An inverted pendulum system, and now also a robot arm emulator has been included in the distribution. The system and on-line documentation is available in the home web page of FC-DT, at http://www-pr.unsl.edu.ar/projects/fc-dt. The reader can reproduce the results shown here just by running the system, and of course, can experiment with new problems.

10 Conclusions

The enhanced version of the evolutionary algorithm presented in this paper can successfully generate fuzzy controllers for the robot arm control problem. The algorithm is very robust because it can obtain successful fuzzy

controllers in every run. The quality of the obtained fuzzy controllers can still be enhanced by running the neural network based algorithm that was proposed in previous works [12] [13].

The incremental learning strategy allows to reduce the learning time, because systems that fail to go in the right direction from the beginning, are discarded as soon as they fail to fulfill the requirements.

The codification proposed is general and can be applied to other problems as well.

The system is freely available allowing other members of the research community to experiment with it.

References

[1] D. E. Moriarty and R. Miikkulainen (1996). *Efficient Reinforcement Learning through Symbiotic Evolution.* Machine Learning, 22:11-32.

[2] D. E. Moriarty and R. Miikkulainen (1996). *Hierarchical Evolution of Neural Networks.* Technical Report AI96-242. Department of Computer Sciences, The University of Texas at Austin.

[3] D. D. Leitch (1995). *A New Genetic Algorithm for the Evolution of Fuzzy Systems.* PhD Thesis. Department of Engineer Science, University of Oxford.

[4] A. G. Barto, R. S. Sutton and C. W. Anderson (1983). *Neuronlike adaptive elements that can solve difficult learning control problems.* IEEE Transactions on Systems, Man, and Cybernetics, SMC-13:834-846.

[5] H. R. Berenji and P. Khedkar (1992). *Learning and Tuning Fuzzy Logic Controllers Through Reinforcements.* IEEE Transactions on Neural Networks, vol. 3, no. 5.

[6] C. L. Karr (1991). *Design of a Cart-Pole balancing Fuzzy Logic Controller using a Genetic Algorithm.* SPIE Conf. on Applications of Artificial Intelligence, WA.

[7] M. A. Lee and H. Takagi (1993). *Neural Networks and Genetic Algorithms Approaches to Auto-Design a Fuzzy System.* FLAI'93. Springer Verlag, Berlin.

[8] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine learning.* Reading, MA, Addison Wesley.

[9] Z. Michalewicz (1994). *Genetic Algorithms + Data Structures = Evolution Programs.* Springer Verlag, 2nd ed.

[10] S. Nolfi, D. Floreano, O. Miglino and F. Mondada (1994). *How to evolve autonomous robots: Different approaches in evolutionary robotics.* Artificial life IV. Cambridge, MA.

[11] P. van der Smagt (1994). *Simderella: a robot simulator for neuro-controller design.* Neurocomputing. Vol. 6 No. 2. Elsevier Science Publishers.

[12] Carlos Kavka, María Liz Crespo and Marcelo Cena (1997). *Definition of load balancing algorithms in distributed systems through neuro-fuzzy systems.* Second International Symposium on Soft Computing SOCO'97, Nîmes, France.

[13] Carlos Kavka, María Liz Crespo y Marcelo Cena (1998). *Fuzzy systems generation through symbiotic evolution.* International Symposium on Engineering of Intelligent Systems EIS'98, Tenerife, España.