An Architecture for Fuzzy Logic Controllers Evolution and Learning in Microcontrollers based Environments

Carlos Kavka, Patricia Roggero and Javier Apolloni

LIDIC
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950
D5700HHW - San Luis - Argentina
Tel: 02652-420823 Fax: 02652-430224
e-mail: {ckavka,proggero,javierma}@unsl.edu.ar

Abstract

The growing number of control models based on combinations of neural networks, fuzzy systems and evolutionary algorithms shows that they represent a flexible and powerful approach. However, most of these models assume that there is enough CPU power for the evolutionary and learning algorithms, which in a large number of cases is an unrealistic assumption. It is usual that the control tasks are performed by small microcontrollers, which are very near to or *embedded* in the plant, with low power, low cost and dedicated to a single task. This work proposes an architecture for evolution and learning in adaptive control, specifically designed to operate in microcontrollers based environments. An evaluation on a simulated temperature control environment is provided, together with details on the current hardware implementation.

Keywords: Neural Networks, Evolutionary Algorithms, Fuzzy Systems, Control.

1 Introduction

It is widely recognized the need to complement the conventional control theory with elements like logic, heuristic and reasoning in order to deal with complex, nonlinear and imprecisely defined processes [7, 12]. In particular, fuzzy logic (FL), neural networks (NN), evolutionary algorithms (EA) and their combinations have been proven effective in this context [5, 6].

Fuzzy logic in control applications can be considered as a generalization of conventional rule based expert systems, where the input and output variables are defined in terms of linguistic values. Usually, the number of rules that are necessary for control applications are reduced by orders of magnitude [12], when compared with the standard approach based on expert systems. However, two problems remain: (1) the definition of the rule base is still a complicated process that depends on the

expert knowledge, and (2) there is no formal procedure to determine the parameters of the resulting fuzzy system.

Neural networks consist of a set of interconnected processing elements that can learn an input-output mapping by modifying its parameters. They can be trained in order to learn the mapping between the plant state and the actuating commands, producing a working controller [14, 5, 3, 2]. However, obtaining training data for a controller is very difficult and expensive in most real environments [7].

The approaches followed by fuzzy logic and neural networks to solve a control problem are clearly different: fuzzy logic can be used when expert knowledge about the problem is available [11, 13], and neural networks are appropriate when there is enough process data. However, both models are used to build non linear systems, with neural networks representing the variables numerically and fuzzy systems representing them symbolically. Models based on a combination of fuzzy logic and neural networks have been proposed, trying to get together the advantages of both approaches: neural networks provide learning capacity and fuzzy systems an explicit representation scheme for knowledge [12].

It is well known the neural network ability to represent the dynamic behavior of physical processes, but learning and structure definition can be very complicated. Evolutionary algorithms are a search technique inspired by natural evolution, which perform a population based search strategy [6]. The possible solutions are codified as individuals in the population that compete and exchange information with others. Evolutionary algorithms are very robust, since they do not depend on gradient information and can deal with problems where input-output mappings are not provided or even explicit objective functions are not available [15]. Evolutionary algorithms have been used successfully to obtain parameters for fuzzy controllers [9, 8] and for neural networks [7, 5, 15].

The growing number of control models based on combinations of neural networks, fuzzy systems and evolutionary algorithms shows that they represent a flexible and powerful approach [1]. However, most of these models assume that there is enough CPU power for the evolutionary and learning algorithms, which in a large number of cases is an unrealistic assumption. It is usual that the control tasks are performed by small microcontrollers, which are very near to or *embedded* in the plant, with low power, low cost and dedicated to a single task. Today, the industry sells ten times as many microcontrollers as microprocessors, making a proper solution based on these new techniques (NN, EA, FL) very appealing. The last generation of microcontrollers introduces a full fledge Ethernet controller as a main component, providing the possibility to design a network based control algorithm.

This work proposes an architecture for evolution and learning in adaptive control, specifically designed to operate in microcontrollers based environments. The two main objectives of this architecture are: (1) the development of a fuzzy logic controller to be executed on a microcontroller in order to control a process (plant), and (2) the automatic detection of changes in the control task and the corresponding adaptation of the fuzzy logic controller in the main computer system to meet the new requirements. Section 2 presents the proposed adaptive control architecture. Section 3 explains the dynamic process modeling using neural networks, section 4 shows details on the fuzzy controller, section 5 gives details on the evolutionary algorithm used and section 6 introduces the quality monitoring system. An example of this architecture is presented in section 7 and the details on the current hardware implementation in section 8.

2 Adaptive Control Architecture

The proposed adaptive control architecture is shown in figure 1. The microcontroller performs the control task by using a fuzzy controller obtained from an evolutionary neural network (ENN). The evolutionary algorithm runs on a computer system that is connected through the network and not on the microcontroller itself. The microcontroller requests evolutionary update of the fuzzy controller when it detects that it is necessary to adjust the parameters of the current controller.

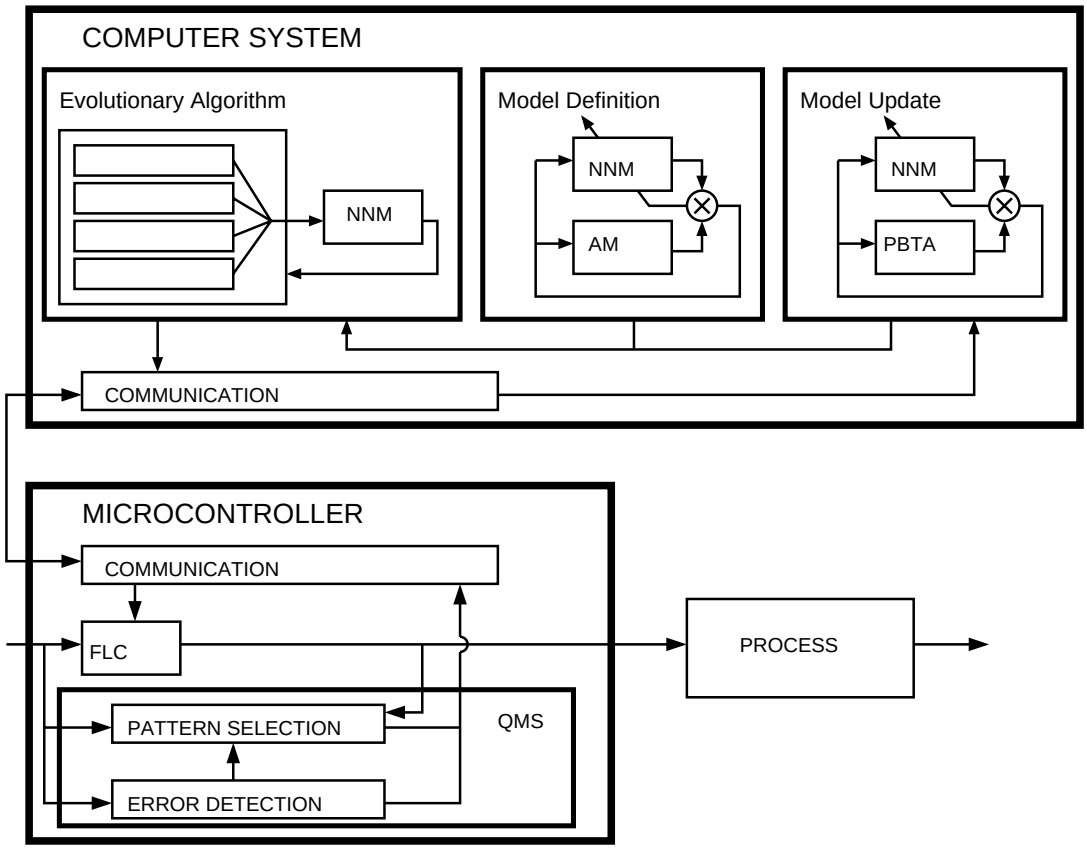

Figure 1: The proposed Adaptive Control Architecture (NNM: neural network model, AM: analytical model, PBTA: pattern based training algorithm, FLC: fuzzy logic controller, QMS: quality monitoring system).

The architecture consists of two main components: the set of modules that run on the main computer system, and the modules that are executed on the microcontroller. The main computer system executes the following modules:

Model Definition : This module builds a neural network that will act as a model of the plant, based on its best analytical representation. The neural network model (NNM) is trained to learn a direct mapping between the plant state and the actuating commands. Details on the operation of this module are presented in section 3.

Evolutionary Algorithm : This module builds a fuzzy logic controller by using evolutionary techniques. Each individual in the population is a controller that is evaluated by considering its performance in the control of the neural network model. The best evolutionary fuzzy logic controller is selected for the control task. Section 5 gives details on this module.

Model Update : This module updates the parameters of the direct neural network model (NNM) by using a pattern based training algorithm (PBTA). This module is executed only when the current fuzzy logic controller exhibits poor performance, due to changes in the non linear properties of the plant. Details on the operation of this module are presented in section 3.

Communication : This module is responsible of the data transfer between the modules in the main computer system, and the modules in the microcontroller.

The microcontroller executes the following modules:

Fuzzy Logic Controller : It is a fuzzy logic controller that performs the actual control task by using the parameters provided by the evolutionary algorithm. Section 4 gives details on it.

Error Detection : This module detects the differences between the expected and the actual behavior of the controller. When the difference goes beyond a certain limit, it triggers the pattern selection mechanism, the update of the direct model and the evolutionary step in order to update the fuzzy logic controller. More details are presented in section 6. This module and the next one together belong to the so called Quality Monitoring System (QMS).

Pattern Selection : This module is responsible for selecting adequate patterns for the retraining of the direct model when the characteristics of the plant have changed in time. Details on the operation of this module are presented in section 3.

Communication : This module is responsible of the data transfer between the modules in the microcontroller, and the modules in the main computer system.

3 The Direct Model

Evolutionary algorithms have been seldom used for real time adaptation of fuzzy logic controllers, since they require a large number of evaluations of the target system. Usually, an accurate simulation model for evaluations is used. This is also the case in our architecture. The model is represented with a neural network that has to learn the mapping between the plant state and the actuating commands, or in other words, it has to become a model of the plant. Since this information is not usually available as a set of input-output patterns, the network is built by performing training with an (approximate) analytical model of the plant developed by a human expert (differential equations, etc.). This task is performed by the Model Definition module in the proposed architecture (see figure 1). The neural network is a standard feed forward network trained with the resilient back propagation algorithm (see figure 2). The input pattern $x(t) = (x_t, x_{t-1}, \ldots, x_{t-p})$ in time t corresponds to the regression vector of the p past input values $x_t, x_{t-1}, \ldots, x_{t-p}$, and the output pattern corresponds to the plant output

$y(t) = (y_t)$ at time t. Note that each x and y can be vectors depending on the number of status variables and plant outputs. This regression model is necessary since usually the plant behavior is not a direct function of its inputs. The selection of p is problem dependent.

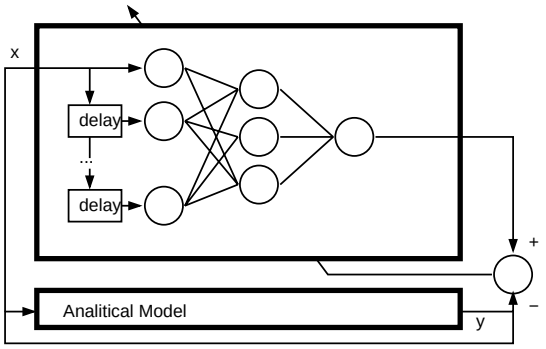

Figure 2: The direct model of the plant represented with a neural network trained with an analytical representation of the target system.

The neural network model can be not enough accurate due to: (1) incorrect definition of the analytical model or (2) changes in the behavior of the plant. In both cases, the situation will be signaled by the Error Detection module, causing a set of patterns to be selected by the Pattern Selection module. The direct neural network model will then be updated with standard learning on this set of patterns by the Model Update module.

4 The Fuzzy Controller

A number of different models of fuzzy systems have been proposed in the literature [1]. The main component in all of them is a rule base, where the main differences are in the definition of the fuzzy reasoning method used. The fuzzy reasoning method corresponds to the inference operations that are performed on the fuzzy rules. In our work, we have selected one of the most successful fuzzy system on control applications: the Takagi-Sugeno fuzzy system [17, 16, 13].

A Takagi-Sugeno fuzzy system has n input variables $x_1, x_2, \ldots, x_n$ and one output variable v. Each input variable x_i is fuzzified by p_i fuzzy sets A_{ij} ($p_i \geq 1$,$1 \leq i \leq n$,$1 \leq j \leq p_i$) whose membership functions μ_{ik_i} ($k_i = 1, 2, \ldots, p_i$) are arbitrarily defined. There must be a rule for each combination of fuzzy sets, and the output is defined as a linear combination of input variables. As an example, the definition of the k-th ($1 \leq k \leq \omega$) rule follows:

$$if\ x_1\ is\ A_1\ and\ \ldots\ and\ x_n\ is\ A_n\ \ then\ v_k = a_{0k} + a_{1k}x_1 + \ldots + a_{nk}x_n$$

where the A_i are the input fuzzy sets for each input variable ($A_i \in \{A_{i1}, \ldots, A_{ip_i}\}$), and a_{0k} and a_{ik} are adjustable real valued parameters.

The first step in the evaluation of the fuzzy system is the computation of the membership functions $\mu_{A_i}(x_i)$ for each input value x_i. As an example, symmetric triangular membership functions with

center m_i and width d_i are defined as follows:

$$\mu_{A_i}(x_i) = \begin{cases} 1 - \frac{|x_i - m_i|}{d_i} & \text{if } |x_i - m_i| < d_j \\ 0 & \text{elsewhere} \end{cases} \quad (1)$$

Then the membership values are combined in a scalar value that corresponds to the rule firing strength $w_k(x)$ with a standard `t-norm` operator, with product and minimum as usual selections. As an example, the product combination is defined as follows:

$$w_k(x) = \mu_{A_1}(x_1) \times \ldots \times \mu_{A_n}(x_n) \quad (2)$$

where $x =< x_1, \ldots, x_n >$ is the complete input vector.

The firing strength of each rule is computed as the ratio of its firing strength to the sum of all rules' firing strengths as follows:

$$\overline{w}_k = \frac{w_k}{\sum_{i=1}^{n} w_i} \quad (3)$$

The output of the fuzzy system is then computed by adding the output value produced by each rule multiplied by its weighted firing strength:

$$v(x) = \sum_{k=1}^{\omega} \overline{w}_k * v_k(x) \quad (4)$$

5 The Evolutionary Algorithm

The evolutionary algorithm evolves a population of individuals, with each one representing a fuzzy controller. All of them are evaluated on the simulated plant (the neural network model), and the best one is selected for the control task.

The fuzzy system is represented by using the Fuzzy Voronoi Representation (FVR), which is an extension of the representation proposed in [10], now used to model the joint fuzzy set defined by the antecedents of the fuzzy rules by using Voronoi diagrams. The advantages of this approach are the simplicity involved in its definition, the possibility it offers to use geometric properties for fuzzy rule manipulation and, since it represents only one fuzzy set for each rule, the reduced computation time. The last one is particulary interesting in a microcontroller based environment.

A Voronoi diagram induces a subdivision of the space based on a set of points called *sites*. An important property is that a number of operations can be executed on its topological structure just by operating with the sites. Formally [4], a Voronoi diagram of a set of n points P is the subdivision of the plane into n cells, with the property that a point q lies in the cell corresponding to a site p_i if and only if the distance between q and p_i is smaller than the distance between q and p_j for each $p_j \in P$ with $j \neq i$. Figure 3 shows an example of a Voronoi diagram in 2D. A related concept is the Delaunay triangulation. A triangulation [4] of a set of points P is defined as the maximal planar subdivision whose vertex set is P. A maximal planar subdivision S is a subdivision such that no edge connecting two vertices can be added to S without destroying its planarity. In other words, any edge that is not in S intersects one of the existing edges. A triangulation $\mathcal{T}$ of a set of points P is a

Delaunay triangulation if and only if the circumcircle of any triangle in $\mathcal{T}$ does not contain a point of P in its interior. A circumcircle of a triangle is defined as the circle that goes through its three summit. Figure 3 shows an example of a Delaunay triangulation in 2D

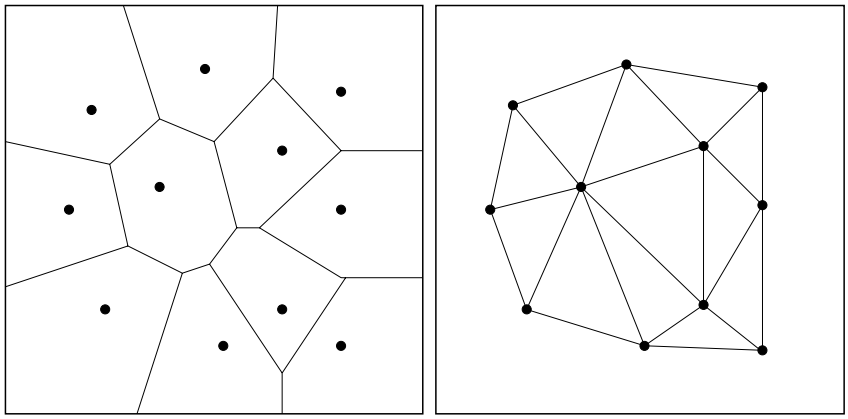

Figure 3: An example of a Voronoi diagram defined by a set of points in 2D (left) and the corresponding Delaunay diagram (right).

By using the FVR representation, each individual contains a variable number of fuzzy rules, where each rule is defined in terms of its joint fuzzy set. As an example, the k-th ($1 \leq k \leq \omega$) rule using the FVR in a Takagi-Sugeno fuzzy system can be defined as follows:

$$if\ x\ is\ A_k\ then\ v_k = a_{0k} + a_{1k}x_1 + \ldots + a_{nk}x_n$$

where A_k is a joint fuzzy set, $x =< x_1, x_2, \ldots, x_n >$ the input vector defined as the concatenation of all input variables x_i ($1 \leq k \leq n$) , and a_{0k} and a_{ik} ($1 \leq i \leq n$) are adjustable real valued parameters.

Each rule defines the joint fuzzy set by just specifying a point in the input domain. This point corresponds to the center of the Voronoi region. A set of fuzzy rules defines a complete Voronoi diagram. Note that with a one dimensional input space, the joint fuzzy set corresponds to a single fuzzy set. In two dimensions, the input space is partitioned as shown in figure 3.

The Voronoi diagram defines just the partition of the input space. The membership value of the input vector x to the joint fuzzy sets A_k can be defined in different ways, depending on the overlapping criteria and the shape of the joint fuzzy set. A useful joint fuzzy set can be defined as follows:

$$\mu_{A_k}(x) = l_1(x)$$

where $l_1(x)$ is the first barycentric coordinate of x in the simplex T defined by the Delaunay triangulation to which x belongs. This coordinate will get the value 1 when x is the center of the region, and it will go down linearly to 0 in the centers of the neighbor joint fuzzy sets. In order to allow this function to be defined everywhere, a very large triangle containing all points in the domain is defined. An example of this joint fuzzy set in FVR is shown in figure 4.

The crossover and mutation operators are defined in terms of geometric operators, as detailed in [10].

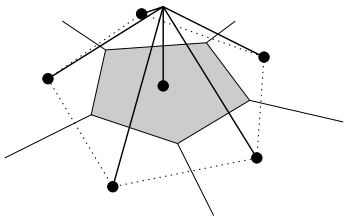

Figure 4: An example of a joint fuzzy set. Membership is maximum at the center of the Voronoi region and it goes down linearly to 0 in the centers of the neighbor Voronoi regions.

6 The Quality Monitoring System

The Quality Monitoring System is composed of two submodules: the Error Detection and the Pattern Generation modules. The function of the Error Detection module is to detect failures in the performance of the fuzzy controller, based on the changes in the status variables in time. It is defined by the human operator and it is clearly dependent on the characteristics of the plant. If it is possible to associate an scalar error $e(t)$ for each state of the plant at time t, then the Error Detecting module triggers when:

$$\frac{1}{N}\sum_{\tau=t-N}^{\tau=t} e(\tau)^2 > \epsilon$$

computed for the last N time steps and for a given ϵ value. In other situations, the error for each step can be defined in terms of differences between the k normalized state variables $x_i(t)$ and specific target values $x_i'(t)$. In this case, the Error Detecting module triggers when:

$$\frac{1}{N}\sum_{\tau=t-N}^{\tau=t}\sum_{j=1}^{k}(x_i(\tau) - x_i'(\tau))^2 > \epsilon$$

The error can be defined also with other methods, like for example a fuzzy system properly designed, as for example in the GARIC model [3].

The Pattern Generation module selects a set of patterns $[x(t), y(t)]$, from the time interval in which the fuzzy controller cannot control the plant with the expected performance. They will be used by the Update Model module to enhance the neural network to be adapted to the new plant reality. Then the evolutionary algorithm will obtain a new controller for it.

7 Simulation Experiment

The proposed architecture was evaluated in a temperature control simulated environment, as shown in figure 5. The Fuzzy Logic Controller can control the temperature of the physical medium by actuating on a heater. The temperature of the medium is measured by a thermometer.

The analytical equation that defines the simulated environment is:

$$T(t) = Ws + \exp^{(-\frac{\alpha t}{c\rho})}(-s + T0)$$

where α is the heat transfer coefficient, ρ is the medium density, c is the specific heat, $T0$ is the initial temperature, s is the control signal and W is the heater power.

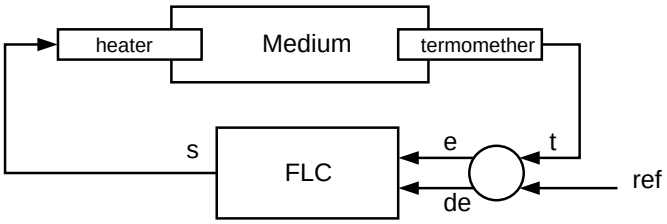

Figure 5: The simulation environment of the temperature control system.

The objective of the control task is to keep the medium at a desired temperature ref, called the reference value. The controller has two inputs: the error e computed as the difference between the reference and the current temperature value, and de which is the derivative of the error, or in other words, the change of the error in time.

An example of the plant behavior is shown on the left plot in figure 6. The x axis corresponds to the time and the y axis to the temperature. The reference value is 50 degrees and the initial temperature $T0$ is 20 degrees. The other parameters are $\alpha = 30$, $\rho = 2200$, $c = 170$ and $W = 1$. A neural network is trained with the resilient back propagation algorithm to emulate the behavior of the plant. The second graph displays the NNM behavior when applied to the same input, showing that it is an approximate model of the plant.

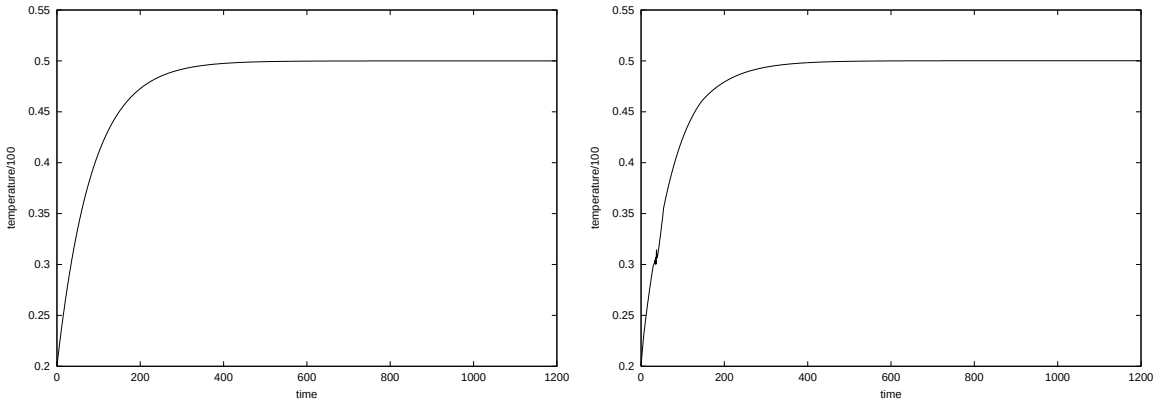

Figure 6: The plant behavior (left) and the neural network model behavior (right) with the same inputs.

The Evolutionary Algorithm is then used to get a fuzzy logic controller. The graph on the left of figure 7 shows the signal the best controller sends to the heater in order to reach the reference temperature. Note that initially this value is high, and then it is lowered in order not to overpass the reference value. The graph on the right of the same figure shows the changes in temperature in the plant.

In order to simulate a physical change in the characteristics of the system, the value of W is reduced by 20%, simulating a reduction of the power of the heater. The graph on the left of figure 8

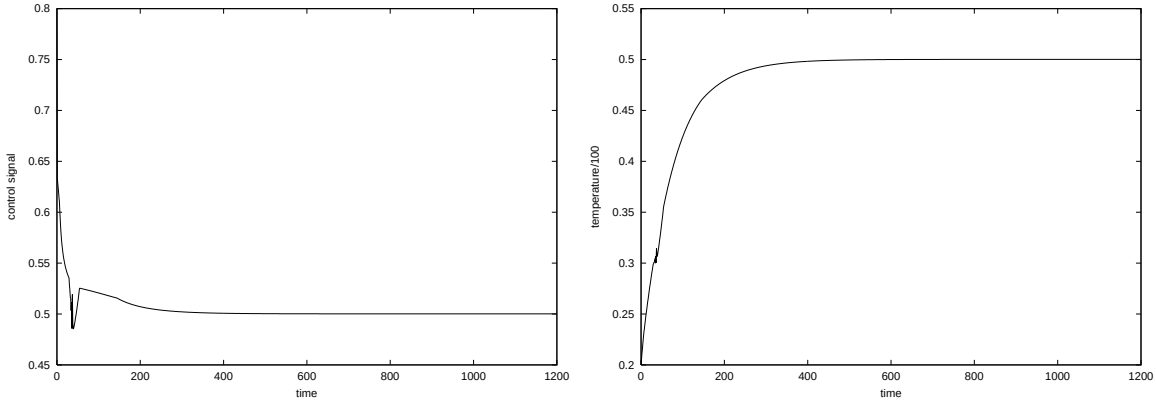

Figure 7: The signal produced by the best fuzzy controller (left) and its corresponding temperature change in the physical medium.

shows that now the controller cannot reach the reference temperature. After a number of time steps, the quality monitoring system discovers the situation and starts the pattern selection. The NNM is re-trained with this new pattern set reflecting the new behavior of the physical system. The evolutionary algorithm is then executed in order to build a new controller. The graph on the right of the figure 8 displays the behavior of the new controller, showing that now it can control the system.

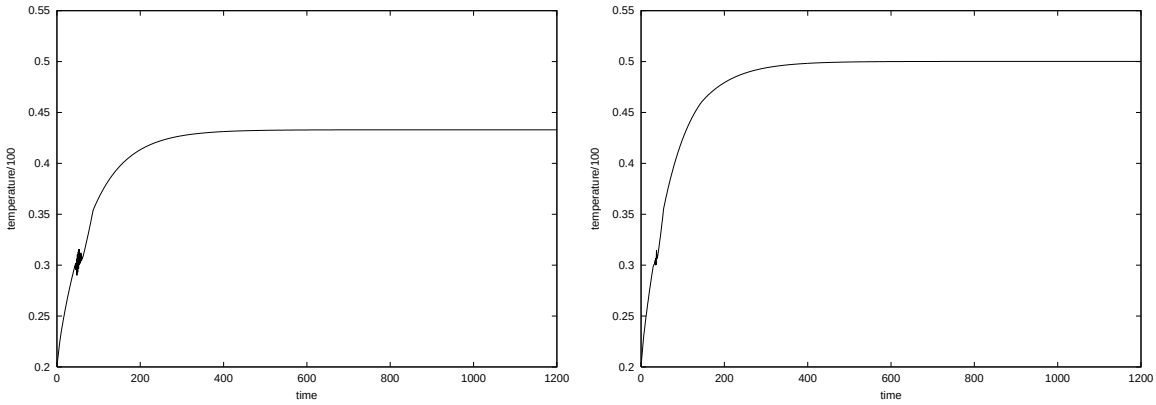

Figure 8: The behavior of the original controller (left) and the new controller (right) on the modified physical system

.

8 Current Hardware Implementation

A temperature control system is currently under implementation in order to evaluate the proposed architecture in a real hardware environment. The microcontroller is a DS80C390 from Maxim semiconductors, with 512K flash and 1MB static RAM, Ethernet connection and a proprietary real time

operating system. A set of digital thermometers and digital potentiometers, together with a heating system based on resistors is being used to build the evaluation system.

9 Conclusions

An architecture for fuzzy logic controllers evolution and learning in microcontroller based environments was proposed. Neural network learning is used to build accurate models of the plant. Evolutionary algorithms are used to build the fuzzy logic controller. A special representation for the individuals was proposed in order to take advantage of geometric properties that showed to be useful in other contexts. In this way, it is possible to generate more efficient fuzzy controllers, making them more adequate to be executed on microcontrollers. The architecture was evaluated in a simulated temperature control environment, showing that it is responsive to changes in the controlled process. Currently a hardware implementation is under study, in order to evaluate the proposed architecture in a real environment.

References

[1] L. Jain A. Abraham and J. Kacprzyk, editors. *Intelligent Systems: Architectures and Perspectives, Recent Advances in Intelligent Paradigms and Applications*, chapter 1, pages 1–35. Studies in Fuzziness and Soft Computing. Springer Verlag Germany, 2002.

[2] A. Abraham. Evonf: A framework for optimization of fuzzy inference systems using neural network learning and evolutionary computation. In *The 17th IEEE International Symposium on Intelligent Control*, Canada, 2002. IEEE Press.

[3] H. R. Berenji and P. Khedkar. Learning and Tuning Fuzzy Logic Controllers Through Reinforcements. *IEEE Transactions on Neural Networks*, 3(5), 1992.

[4] M. de Berg M. van Kreveld M. Overmars and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications*. Springer-Verlag, second edition, 1998.

[5] Emile Fiesler and Russel Beale, editors. *Handbook of Neural Computation*. IOP Publishing Ltd and Oxford University Press, 1997.

[6] Thomas Back David Fogel and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Algorithms*. IOP Publishing Ltd and Oxford University Press, 1997.

[7] Athula Rajapakse Kazuo Furuta and Shunsuke Kondo. Evolutionary learning of fuzzy logic controllers and their adaptation through perpetual evolution. *IEEE Transactions on Fuzzy Systems*, 10(3):309–321, June 2002.

[8] F. Hoffmann. Evolutionary algorithms for fuzzy control system design. *Proceedings of the IEEE, Special Issue on Industrial Innovations using Soft Computing*, pages 1318–1333, 2001.

[9] Homaifar and E. D. McCormick. Simultaneous design of membership functions and rule sets fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(1):129–138, May 1995.

[10] Carlos Kavka and Marc Schoenauer. Voronoi based function approximation. In *Genetic and Evolutionary Computation Conference GECCO*, Chicago, USA, 2003.

[11] Bart Kosko. Fuzzy systems as universal approximators. *IEEE Transactions on Computers*, 43(11):1329–1333, 1994.

[12] D. A. Linkens and H.O. Nyongesa. Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications. *IEE Control Theory Applications*, 143(4):367–386, July 1996.

[13] S. Mitaim and B. Kosko. The shape of fuzzy sets in adaptive function approximation. *IEEE Transactions on Fuzzy Systems*, 9(4):637–656, 2001.

[14] Junhong Nie and D. A. Linkens. Learning control using fuzzified self-organizing radial basis function network. *IEEE Transactions on Fuzzy Systems*, 1(4):280–287, November 1993.

[15] Zin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, September 1999.

[16] H. Ying. Constructing nonlinear variable gain controllers via the takagi-sugeno fuzzy control. *IEEE Transactions on Fuzzy Systems*, 6(2):226–234, 1998.

[17] H. Ying. General siso takagi-sugeno fuzzy systems with linear rule consequent are universal approximators. *IEEE Transactions on Fuzzy Systems*, 6(4):582–587, 1998.