A Study of Prior Knowledge Insertion in Evolutionary Fuzzy Recurrent Controllers Design

Javier Apolloni, Carlos Kavka and Patricia Roggero

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: {javierma,ckavka,proggero}@unsl.edu.ar

Abstract

A fuzzy controller is usually designed by formulating the knowledge of a human expert into a set of linguistic variables and fuzzy rules. As it is well known, the use of prior knowledge can dramatically improve the performance and quality of the fuzzy system design process. In previous works we have introduced the RFV model, a representation for recurrent fuzzy controllers based on Voronoi diagrams that can represent fuzzy systems with synergistic rules, fulfilling the completeness property and providing a simple way to introduce prior knowledge. In this work we present our current approach in the study of the inclusion of prior knowledge in the context of the RFV model.

Keywords: Genetic algorithms, Recurrent fuzzy systems, Fuzzy control, Voronoi diagrams, Evolutionary robotics, Prior knowledge insertion.

1 Introduction

The development of controllers by using fuzzy logic techniques has been subject of fundamental research with many successful applications produced during last years [1]. The main reason is that fuzzy logic controllers (FLC) provide satisfactory performance in face of uncertainty and imprecision [6], while keeping an equivalence in knowledge representation with other methods like neural networks and automata [5]. An FLC represents a non linear model as the combination of a set of local linear models, where each one represents the dynamics of a complex system in a single local region [4]. Each local model is specified by a fuzzy rule, which defines the local region in which the rule applies through the membership functions used in the antecedent, while the consequent defines the output of the model.

One of the advantages of fuzzy systems is the possibility to use prior knowledge, a well known concept that specifies the information about the desired form of a solution which is additional to the information provided by the training data [2]. The correct use of prior knowledge during model design leads to better models even in the presence of deficient and incomplete data sets [12].

Most non linear problems in control require the processing of temporal sequences, or in other words, in these problems the output depends on the current input and previous values of inputs and/or

outputs. Fuzzy recurrent models can deal with this kind of problems. However, even if these recurrent models are successful in supporting learning of temporal sequences, in most cases the logic interpretation of recurrent units is not considered. In previous works, we have proposed a recurrent structure for fuzzy systems based on the Fuzzy Voronoi (FV) method proposed in [9], that allows the definition of recurrent fuzzy systems with a clear interpretation of recurrent units. The Recurrent Fuzzy Voronoi (RFV) presented in [7] structure consists in a set of rules, where the antecedent of the rules are determined by multidimensional membership functions defined in terms of Voronoi regions. The RFV model includes external and internal variables with recurrent connections that allow the processing of temporal sequences of arbitrary length. Evolutionary algorithms are proposed as a design tool, since they do not require derivative information, which in most control problems is unavailable or costly to obtain.

In this paper, we extend the study of the prior knowledge inclusion properties of the RFV model by proposing two modifications to the evolutionary design approach. As a first step, free recurrent units with no definite semantic are included into the fuzzy system in order to provide more freedom degrees to the search algorithm. As a second step, we allow the consequents of the prior fuzzy rules to be included in the evolutionary process, in order to fine tune the prior defined knowledge.

This paper is organized as follows. Section 2 describes the structure of the RFV model. In section 3, the design of fuzzy controllers by using genetic algorithms is introduced. Section 4 presents the extensions proposed to the RFV design. Section 5 presents experimental results and finally, section 6 presents the conclusions.

2 The RFV model

In this section, the structure of the RFV model is presented, the fuzzy reasoning strategy is explained and the details on the computation of the membership functions are introduced.

2.1 Structure

A schematic diagram of the model is shown in figure 1, which is organized in four layers and consists of l input variables, r internal variables, m output variables and ω rules.

Units in layer 1 are called input units. There are two types of input units: external inputs and internal units that are used also as standard inputs in rule definition. Units in layer 2 are called partition units. They act as multidimensional fuzzy membership functions. Units in layer 3 are called rule units. Each fuzzy rule in the fuzzy system has a corresponding rule unit. There is a one to one correspondence with units in layer 2. Units in layer 4 are called output units. They compute the outputs as a weighted linear combination of input units, generating both the external outputs and the values of the internal units to be made available as inputs in the next time step. The internal units implement the recurrent connections of the model.

The function of each type of unit is described below. In the descriptions, the external input vector of size l is denoted by x, the internal vector of size r is denoted by h, the output vector of size m is denoted by y, the complete input vector for the rules of size $l + r$ is denoted by $I = x : h$ and the complete output vector produced by the rules of size $m+r$ is denoted by $O = v : y$, where : identifies the concatenation operator.

Layer 1 : No computation is performed in this layer. External input values x and previous values of internal units y are transmitted to the units in layer 2.

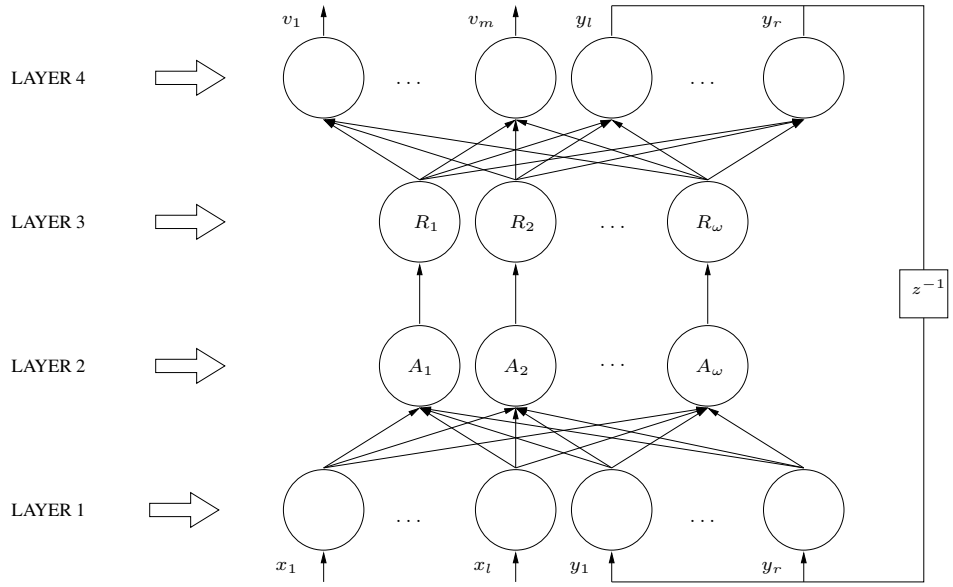


Figure 1: The structure of the RFV model

Layer 2 : The k-th unit in this layer computes the fuzzy membership value $\mu_{A_k}(I)$ of the input vector I to the multidimensional fuzzy set A_k associated to the k-th rule. More details on this computations are provided in section 2.2.

Layer 3 : Units in this layer compute a linear combination of input values based on the parameters specified by each rule, weighted by the corresponding degree of activation, as usual in TS fuzzy systems. Note that this units produce $m + r$ outputs. The output produced by the unit k that corresponds to the output variable i is:

$$O_i^k = (a_{k0}^i + \sum_j a_{kj}^i I_j)\mu_k(I) \tag{1}$$

where the a_{kj}^i are the real valued parameters that compute the linear combination of input values associated to the rule k for the output variable i.

Layer 4 : Units in this layer compute the output vector O by computing the summation of the corresponding outputs produced by each rule. That is:

$$O_i = \sum_k O_i^k \tag{2}$$

2.2 Membership computation

The domain partition strategy is based on Voronoi diagrams. A Voronoi diagram induces a subdivision of the space based on a set of points called *sites*. Formally [3], a Voronoi diagram of a set of p points $\mathcal{P} = \{P_1, \ldots, P_p\}$ is the subdivision of the plane into p cells, one for each site in $\mathcal{P}$, with the property that a point M lies in the cell corresponding to a site P_i if and only if the distance between M and P_i is smaller than the distance between P and all other P_j ($j \neq i$).

A related concept is the so called Delaunay triangulation $\mathcal{T}$, defined as the maximal planar subdivision (i.e. a subdivision such that no edge connecting two vertexes can be added to S without destroying its planarity) whose vertex set is $\mathcal{P}$ and such that the circumcircle of any triangle in T does not contain any point of $\mathcal{P}$ in its interior. Figure 2 illustrates an example of a Voronoi diagram and

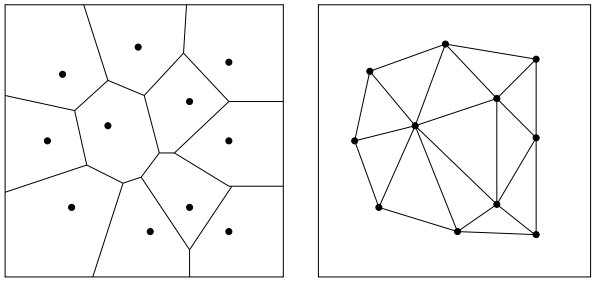


Figure 2: An example of a Voronoi diagram (left) and the corresponding Delaunay triangulation (right) for a set of points in $\mathbb{R}^2$

its corresponding Delaunay triangulation in $\mathbb{R}^2$. Note that these definitions can be straightforwardly extended to $\mathbb{R}^n$, with $n \geq 2$ – all details can be found in [3].

The FV representation [9] considers joint fuzzy sets defined from a Voronoi diagram with p sites $\mathcal{P} = \{P_1, \ldots, P_p\}$. There are as many rules as Voronoi sites. The fuzzy set S_k is defined as by its multivariate membership function μ_k that takes its maximum value 1 at site P_k, and decreases linearly to reach value 0 at the centers of all neighbor Voronoi sites. An example of such a joint fuzzy set is shown in figure 3-a for $n = 2$.

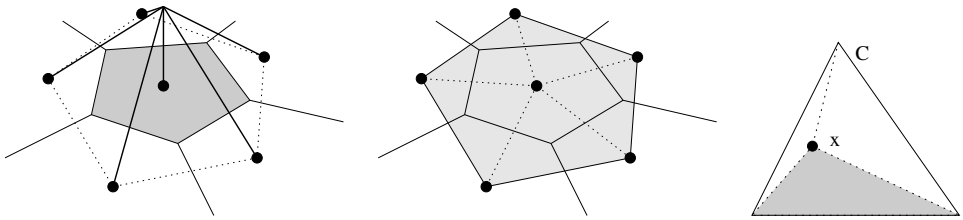


Figure 3: An example of a (a) joint fuzzy set for a single Voronoi region for $n = 2$, where the membership value is represented in the z-axis, and a (b) Voronoi diagram (solid line) and its corresponding Delaunay triangulation (dotted line) for $n = 2$. The graphic (c) shows an example of the membership computation for $n = 2$. The outer triangle corresponds to the simplex defined by the Delaunay triangulation to which x belongs. The membership value corresponds to the area of the shadowed triangle. Note that the value of the area is 1 when x is equal to C and it goes down linearly to 0 on the side of the triangle opposite to C

Formally, the membership value of the input vector I to the joint fuzzy set S_k is defined by:

$$\mu_{S_k}(I) = \begin{cases} l_C(I) & x \in V_k \\ 0 & \text{elsewhere.} \end{cases} \tag{3}$$

where $C = P_k$ is the Voronoi site defining S_k and the Voronoi cell V_k, and $l_C(I)$ is the barycentric coordinate of I in the simplex $T_C(I)$ of the Delaunay triangulation of $\mathcal{P}$ that has C as a vertex and to which I belongs. Figure 3-b shows an example of the Voronoi diagram and the associated Delaunay triangulation. On Figure 3-c, the barycentric coordinate $l_C(I)$ corresponds to the (normalized) gray area (volume if $n > 2$) of the sub-simplex formed by I and vertexes of simplex $T_C(I)$ but C. Note that a very large triangle containing all points in the domain is defined in such a way that there are no open Voronoi regions in the input domain.

The Voronoi based domain partition strategy contrasts with the standard grid partition used in most fuzzy systems, where the input space is divided regularly by considering the domain intervals defined by the linguistic terms associated to each input variable. The grid approach suffers from the

curse of dimensionality, while a multidimensional partition strategy, like the one proposed here, can define regions with a size that depends on the application area of the rules trying to exploit the local complexity of the problem being modeled.

3 RFV design with evolutionary algorithms

Evolutionary algorithms are selected as the optimization tool for RFV controller design, since they have been very successful on problems where training data or gradient information is very difficult or costly to obtain, like most control problems. A floating point coding scheme is selected, where each individual (or chromosome) represents all free parameters of the RFV controller as a variable length vector of floating point values. An individual I with ω rules is defined as the vector:

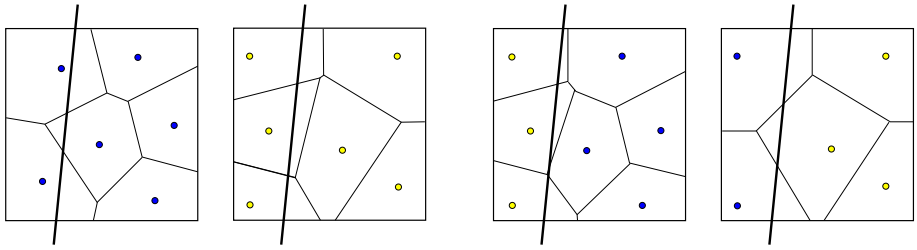
$$Ind = R_1 : \ldots : R_\omega \tag{4}$$

where each sub-vector $R_i (1 \leq i \leq \omega)$ is defined as the floating point vector:

$$R_i = P_i^1 : \ldots : P_i^{l+r} : a_{i0}^1 : \ldots : a_{i(l+r)}^{m+r} \tag{5}$$

where the P_i^j are the coordinates of the site and a_{kj}^i are the real valued parameters associated to the rule R_i.

The evolutionary algorithm is described in details in [11, 8]. The crossover operator is based on geometrical exchange of Voronoi sites between two parents with respect to a random hyperplane. In a problem with input space of dimension d, a random hyperplane h of dimension $d-1$ is randomly defined. The first child receives the local vectors from the first parent that lie on the left of h, and the local vectors from the second parent that lie on the right of h. The second child receives the remaining local vectors. Figure 4 presents an example of the application of this operator in $\mathbb{R}^2$. The mutation operator can either modify the parameters of a particular rule by some standard Gaussian mutation, or add or delete a Voronoi site, i.e. a rule. Practical details on the algorithms, including all parameters, will be given in section 5.



(a) the two parents (b) the two children

Figure 4: An example of the application of the Voronoi crossover in $\mathbb{R}^2$. A random line is drawn of the input space partition of both parents (a), and the Voronoi regions (plus the associated rule parameters) are exchanged, leading to the two children (b).

4 Prior knowledge insertion

In most fuzzy systems, the user can incorporate a priori knowledge by manually defining fuzzy sets and the corresponding fuzzy rules. This process implies that some restriction on the output values and the partition of the input space is introduced in the evolutionary process, but the expected benefit is that

the evolutionary process, biased toward hopefully good parts of its search space, will converge faster to better solutions. Similarly, the RFV representation allows the definition of prior rules, i.e. fixed Voronoi sites that will not be modified by evolution. But one big advantages of the RFV representation is that the expert does not need to specify the application area of such rules: thanks to a synergistic effect between the rules, the evolutionary process, by adding rules more or less close to the prior rules will also tune its domain of application.

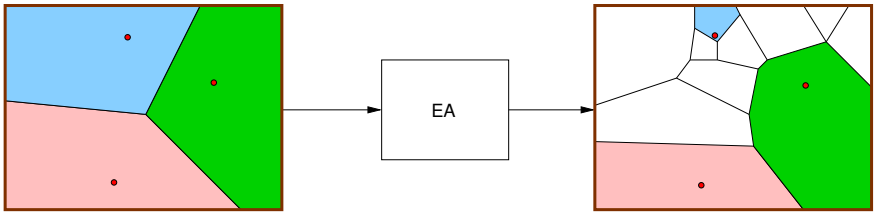


Figure 5: The prior knowledge is defined by specifying a set of rules defined through points in the input domain with a predefined behavior on these points (left). A new set of rules is added (right) by the evolutionary algorithm.

Figure 5 shows an example of the approach. A set of prior fuzzy rules specified by the coordinates of the sites and the real valued parameters associated to each one of them is provided as the initial prior knowledge. This set of rules is included in all individuals, providing a kind of default behavior in the points specified by the rules. The evolutionary algorithm operates by adding rules, which are subject to the effect of variational operators, generating individuals that contains the prior rules plus the new rules obtained by evolution.

Formally, the priori knowledge is defined with a set of m local approximators:

$$Prior = Q_1 : \ldots : Q_m \tag{6}$$

and each individual of size ω, with $\omega > m$ that takes part in the evolutionary process has their first local approximators defined by the prior knowledge:

$$Ind = R_1 : \ldots : R_\omega \tag{7}$$

where $R_i = Q_i, (1 \leq i \leq m)$.

In this approach, the first m local approximators are not modified by variational operators during the evolutionary process. This knowledge insertion method was evaluated in evolutionary robotics in previous works [9] [7]. Next sections present two new approaches to deal with priori knowledge insertion. In the first one, extra free internal units are added to increase the search abilities of the evolutionary algorithms, in the second one, a limited evolutionary fine tuning of prior rule consequents is allowed.

4.1 Free internal units

In the standard knowledge insertion process in the RFV model, the default knowledge is introduced in terms of sites and the behavior to be applied in these points. The prior rules Q_i define the coordinates of the sites and the real valued parameters used to compute the outputs. The outputs include both external and internal units. As it will be shown in next section, the definition of the outputs of internal units with prior rules is important since it allows to provide a semantic for them. However, limiting effects can be introduced in the evolutionary process.

As an extension to the introduction of prior knowledge in the evolution of recurrent controllers, the use of free internal units is proposed. Free internal units are recurrent units which real valued parameters are not defined by prior rules, and are subject to the effect of variational operators. Formally, the parameters a_{kj}^i associated to the free internal units are not specified by the prior rules Q_i, and are affected by mutation operators.

4.2 Limited priori knowledge tuning

In the standard knowledge insertion process in the RFV model, the default knowledge is not modified during the evolutionary process. The rationale behind this approach is that it is expected that the developer has included knowledge that he/she knows that is properly defined. However, it can be not always the case, as experiments will demonstrate on next sections. Generally, the antecedents of the rules, i.e., the area of application of a specific behavior can be determined precisely, however, the exact control signal that has to be applied in these specific points can be difficult to precise.

In the so called limited prior knowledge tuning approach, mutation operators are applied to the consequents of the prior knowledge rules. Formally, the real valued parameters a_{kj}^i that define the consequents of the prior rules Q_i are subject of mutation effects. In this way, the prior rules go under a fine tuning process which can enhance the controller behavior.

5 Experiments

This section presents experimental results of the application of the two proposed approaches for knowledge inclusion in an evolutionary robotic problem. As a test base for experiments, a simulated Khepera robot was used for experimentation. A Khepera robot has 8 sensors that can be used to measure proximity of objects and ambient light levels, and two independent motors to control the speed and direction of the robot. The problem consists in driving the robot while avoiding collisions, starting from a fixed initial position, to a target position that depends on light based signals that are set to on or off status in the trajectory. The presence of an illuminated signal (on status) indicates to the robot that it has to turn left in the next intersection, and its absence (or off status) that it has to turn right. The controller needs internal memory, since the light signal is not present in the intersection, but in a previous (and maybe distant) point in the trajectory. The controller has to learn also when to *forget* light signals that affected the behavior in previous intersections and have not to be considered in other point of the trajectory.

The fitness of a controller is computed in a similar way as in [10] and [7]., evaluating the controller in *e* different scenarios. Each scenario defines initial and target positions, and include path intersections where light signals determine the expected trajectory of the robot. The fitness is accumulated at every step of the robot proportionally to the speed, inversely proportional to the distance to the target point and reduced when the robot travels near obstacles, in order to favor navigation without collisions. The fitness accumulation is stopped when the robot bumps an obstacle, or it reaches a maximum number of steps *s*. The total fitness is the average of the values obtained in the *e* scenarios. Formally, the fitness is defined as follows:

$$\text{fitness}(I) = \frac{1}{es} \sum_{i=1}^{e} \sum_{t=1}^{s} v(t) * (1 - a(t)) * (1 - d(t)) \tag{8}$$

where t is the time step, $v(t)$ is the normalized forward speed (summation of the speed of both motors), $a(t)$ is the normalized maximum activation of the sensors [10] (for example, $a(t) = 1$ implies a collision) and $d(t)$ is the normalized distance to the destination point (for example, $d(t) = 0$ implies

that the target has been reached). This function assigns larger values to individuals that travel at the highest speed, in a trajectory that follows (when possible) a straight line, as far as possible to obstacles and minimizing the distance to the target point.

The evolutionary algorithm for the evolution of the recurrent Voronoi based fuzzy individuals is run with the population size set to 50, the probability of Voronoi crossover set to 0.8, the mutation rate to 0.5, with the perturbation relative mutation set to 0.8, and relative mutation for addition and removal of Voronoi sites set to 0.1 each one. Selection is performed by tournament, no elitism is used and the maximum number of generations is set to 200. The performance of the individuals is measured in $e = 4$ scenarios with three intersections and all combinations of light signals, evaluated in at most $s = 500$ time steps. Figure 6 shows the four training scenarios.

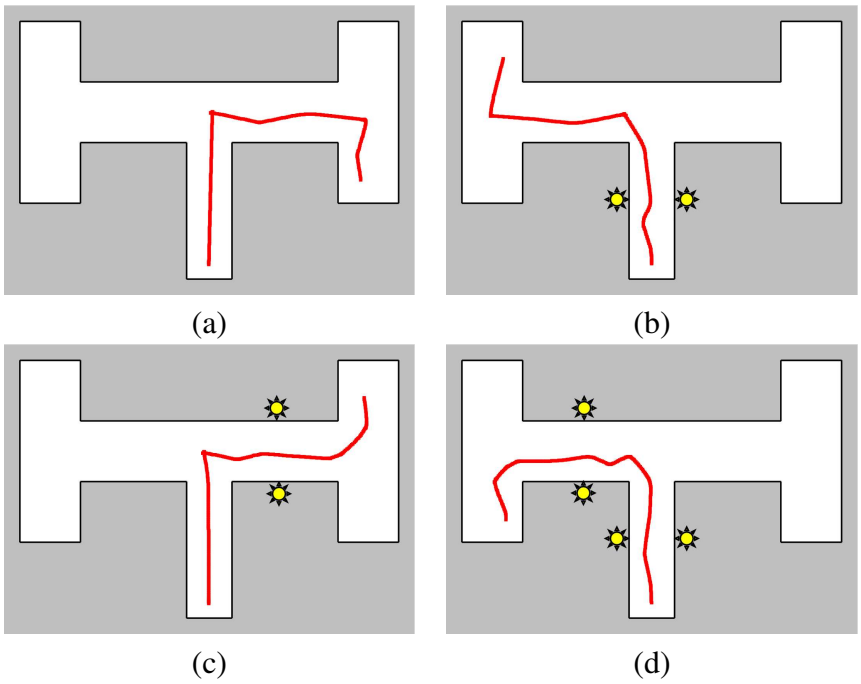


Figure 6: The four scenarios used to evaluate the performance of the controllers. In scenario (a) the robot has to run right in both intersections, in (b) the robot has to turn left first and then right, in (c) the robot has to turn right and then left and in (d) to the left in both intersections.

In the first experiment, the controllers are defined with five inputs, two outputs and one internal variable. The inputs are, respectively, the average of the two left sensors, the two front sensors, the two right sensors, the two back sensors and an average of ambient light as measured by all sensors. The outputs correspond to the speed of the two motors. Note that the presence of an internal variable forces the rules to be defined with six inputs and three outputs (see figure 1). The experiments were performed with prior knowledge defined through the set of rules defined in table 1 and inserted as explained in [9].

The semantic of the prior rules is defined by considering the internal variable y_1 as a flag that indicates if a light signal was seen before. The first four rules correspond to the situation where there are no obstacles near the robot (all distance sensor values are equal to 0). The output produced in all cases for the motors is maximum forward speed (note the constant term of the approximator is 1 for both motor outputs v_1 and v_2). The value of the internal variable y_1 is set to 1 when light is present (rules 3 and 4) and to the previous value (can be 0 or 1) if no light is measured (rules 1 and 2). Rule 5 produces a turn to the right (left motor at maximum speed) if no light was detected before ($y_1 = 0$) and rule 6 a turn to the left (right motor at maximum speed) if light was detected ($y_1 = 1$). In both cases, the flag (internal variable y_1) is reset to 0. It is important to note that the a priori knowledge is

Rule	*site*	v_1	v_2	y_1
	L,C,R,B,G,y_1	a_0^1,a_1^1,a_2^1,a_3^1,a_4^1,a_5^1,a_6^1	a_0^2,a_1^2,a_2^2,a_3^2,a_4^2,a_5^2,a_6^2	a_0^3,a_1^3,a_2^3,a_3^3,a_4^3,a_5^3,a_6^3
R_1	0,0,0,0,0,0	1,0,0,0,0,0,0	1,0,0,0,0,0,0	0,0,0,0,0,0,0
R_2	0,0,0,0,0,1	1,0,0,0,0,0,0	1,0,0,0,0,0,0	1,0,0,0,0,0,0
R_3	0,0,0,0,1,0	1,0,0,0,0,0,0	1,0,0,0,0,0,0	1,0,0,0,0,0,0
R_4	0,0,0,0,1,1	1,0,0,0,0,0,0	1,0,0,0,0,0,0	1,0,0,0,0,0,0
R_5	0,1,0,0,0,0	1,0,0,0,0,0,0	0,0,0,0,0,0,0	0,0,0,0,0,0,0
R_6	0,1,0,0,0,1	0,0,0,0,0,0,0	1,0,0,0,0,0,0	0,0,0,0,0,0,0

Table 1: Prior rules. The value of the site corresponds to the center of the Voronoi region defined by the rule. It is specified by the normalized values of the left (L), center (C), right (R), back (B) and light (G) sensors, and the internal variable y_1. The values a_j^i correspond to the parameters used to define the approximators.

defined by specifying rules that determine the expected behavior of the controller in specific points in the input domain, without specifying the area of application of the rules.

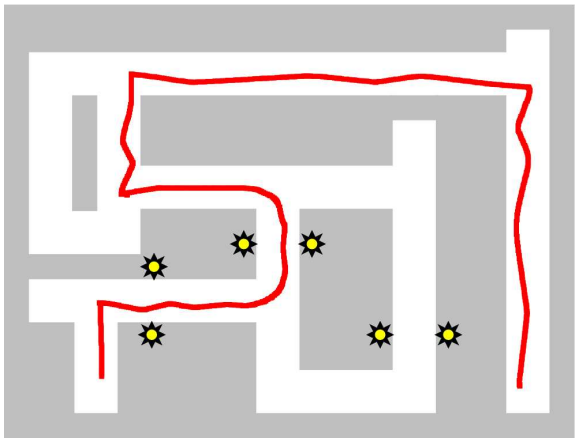
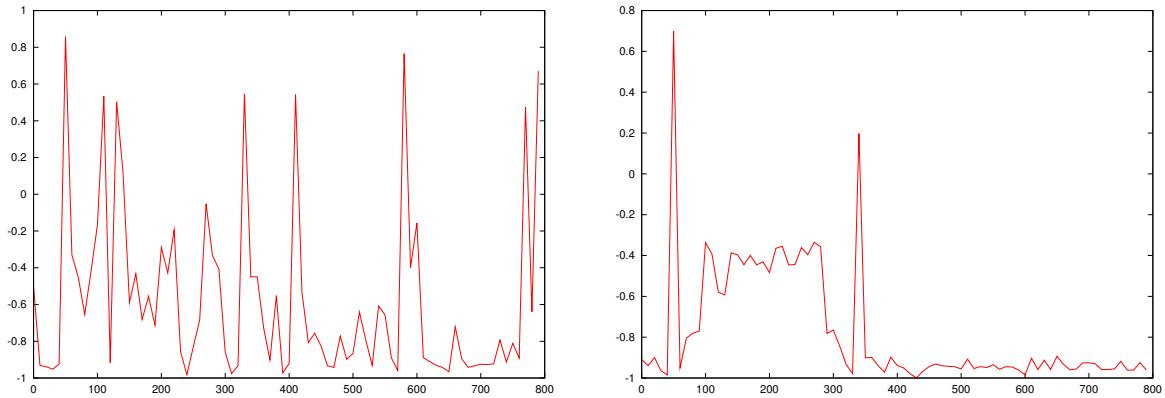


Figure 7: The performance of the best controller when evaluated in a scenario not used during evolution.



(a) (b)

Figure 8: Value of the internal unit of the best controller obtained through evolution (a) without and (b) with prior knowledge when evaluated on the test scenario of figure 7.

The most important point is that a definite semantic interpretation of the internal unit is provided

with the prior rules: the internal unit behavior indicates if light was or not detected before the intersection. There is no guarantee that a clear semantic is provided with the approach without prior knowledge. Figure 8 shows the value of the internal unit of the best controllers plotted when evaluated on the test scenario from figure 7.

The value of the internal unit for the controller evolved with prior knowledge represents the expected semantics, with two peaks on the areas where light signals were detected. This behavior was observed in most controllers obtained after evolution. No clear semantics can be defined in the case of the controller evolved without a priori knowledge.

5.1 Free internal units

The prior knowledge rules establish a behavior on the intersections that forces the robot to turn right or left at maximum speed, producing an abrupt turn. An experiment performed by adding a single extra recurrent unit to the same RFV model used in the base experiment shows that a controller that provides robots with smoother behavior can be obtained through evolution. The prior rules are not modified, but the addition of the extra unit provides more degrees of freedom to the evolutionary algorithm to define controllers that get higher fitness. Figure 9 shows the behavior of the controller in the four training scenarios.

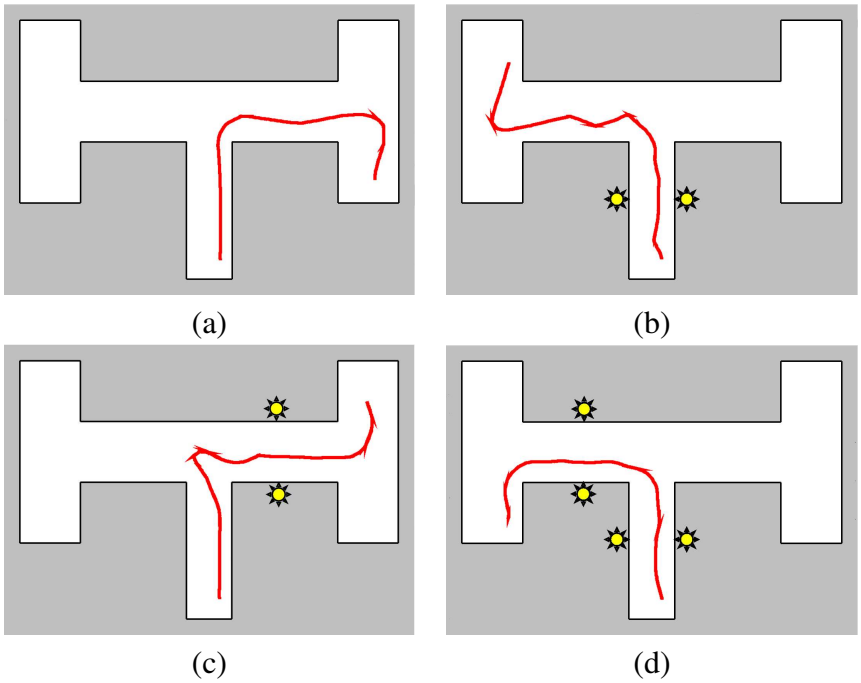


Figure 9: The four scenarios used to evaluate the performance of the controllers with an extra free recurrent unit.

5.2 Limited prior knowledge tuning

In the limited prior tuning experiments, the controllers are defined as in the previous experiments, with the prior knowledge being incorporated through the same set of rules shown in table 1. However, as was detailed in section 4.2, the consequents of the rules are evolved, i.e., the values of the parameters that define the outputs v_1 and v_2 and the internal variable y_1 are modified by mutation during the evolutionary process.

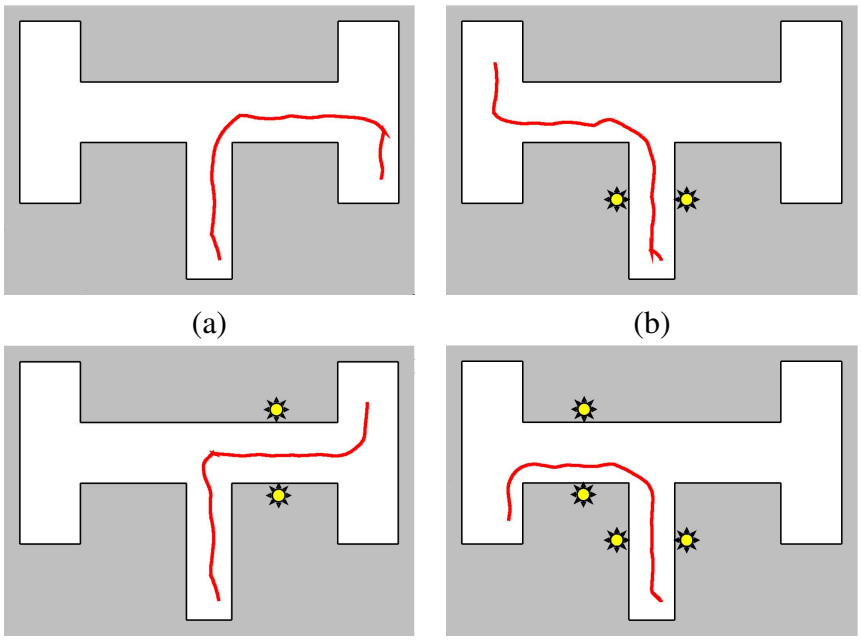


Figure 10: The four scenarios used to evaluate the performance of the controllers with limited prior knowledge tuning.

Figure 10 shows the results obtained on the four training scenarios with the best individual when limited prior knowledge tuning is applied. The antecedents of the prior rules are not modified, however adaption of the consequents is permitted. By comparing the figures 6 and 10 it can be observed that the use of the limited prior knowledge tuning allows to get controllers where the motion of the robot is smoother, reducing the velocity of the motors when the robot must turn right or left. Figure 11 shows the behavior of the best robot in the evaluation scenario.

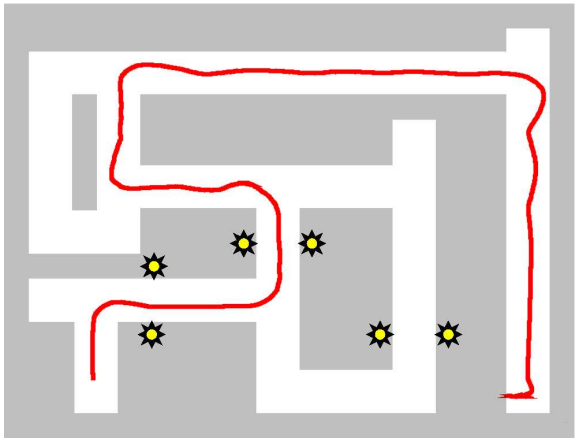


Figure 11: The performance of the best controller obtained with limited priori knowledge tuning when evaluated in a scenario not used during evolution.

6 Conclusions

The RFV model allows the representation of temporal relations by including internal units that provide feedback by connecting outputs to inputs. The main benefit of the model is the possibility to provide

a definite semantic to the internal (or recurrent) units and the possibility to include prior knowledge. Knowledge insertion is important in order to provide a guide in the evolutionary design process. This paper discusses two extensions to the prior knowledge insertion process. In the first one, extra free internal units are added to increase the search abilities of the evolutionary algorithms, in the second one, a limited evolutionary fine tuning of prior rule consequents is allowed. The two approaches were evaluated in a problem in the evolutionary robotics area showing that enhancement to prior knowledge can be useful in some cases.

References

[1] R. Babuška. Fuzzy modeling: Principles, methods and applications. In C. Bonivento, C. Fantuzzi, and R. Rovatti, editors, *Fuzzy Logic Control: Advances in Methodology*, pages 187–220. World Scientific, 1998.

[2] Christopher Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

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

[4] Gang Feng. An approach to adaptive control of fuzzy dynamic systems. *IEEE Transactions on Fuzzy Systems*, 10(2):268–275, April 2002.

[5] C. Lee Giles, Christian W. Omlin, and Karvel K. Thornber. Equivalence in knwoledge representation: Automata, recurrent neural networks and dymical fuzzy systems. *Proc. of the IEEE*, 87(9):1623–1640, September 1999.

[6] Hani A. Hagras. A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Transactions on Fuzzy Systems*, 12(4):524–539, 2004.

[7] Carlos Kavka, Patricia Roggero, and Marc Schoenauer. Evolution of voronoi based fuzzy recurrent controllers. *GECCO Proceedings, ACM*, 2005.

[8] Carlos Kavka and Marc Schoenauer. Voronoi diagrams based function identification. *GECCO, Lecture Notes in Computer Science*, 2723:1089–1100, 2003.

[9] Carlos Kavka and Marc Schoenauer. Evolution of voronoi based fuzzy controllers. *PPSN, Lecture Notes in Computer Science*, 3242:541–550, 2004.

[10] S. Nolfi and D. Floreano. *Evolutionary Robotics, The Biology, Intelligence, and Technology of Self-Organizing Machines*. Bradford Books, 2000.

[11] M. Schoenauer, F. Jouve, and L. Kallel. Identification of mechanical inclusions. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*. Springer Verlag, 1997.

[12] Biing-Tsair Tien and G. Van Straten. The incorporation of qualitative information into t-s fuzzy model. In *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society*, pages 148–153. NAFIP, 1997.