

Evolution of Voronoi based Fuzzy Recurrent Controllers

Carlos Kavka
Departamento de Informatica
Universidad Nac. de San Luis
Ejercito de los Andes 950
D5700HHW, San Luis,
Argentina
ckavka@unsl.edu.ar

Patricia Roggero
Departamento de Informatica
Universidad Nac. de San Luis
Ejercito de los Andes 950
D5700HHW, San Luis,
Argentina
proggero@unsl.edu.ar

Marc Schoenauer
Équipe TAO, INRIA Futurs
LRI
Université de Paris Sud
91405, Orsay Cedex
France
Marc.Schoenauer@inria.fr

ABSTRACT

A fuzzy controller is usually designed by formulating the knowledge of a human expert into a set of linguistic variables and fuzzy rules. Among the most successful methods to automate the fuzzy controllers development process are evolutionary algorithms. In this work, we propose the Recurrent Fuzzy Voronoi (RFV) model, a representation for recurrent fuzzy systems. It is an extension of the FV model proposed by Kavka and Schoenauer that extends the application domain to include temporal problems. The FV model is a representation for fuzzy controllers based on Voronoi diagrams that can represent fuzzy systems with synergistic rules, fulfilling the ϵ -completeness property and providing a simple way to introduce a priori knowledge. In the proposed representation, the temporal relations are embedded by including internal units that provide feedback by connecting outputs to inputs. These internal units act as memory elements. In the RFV model, the semantic of the internal units can be specified together with the a priori rules. The geometric interpretation of the rules allows the use of geometric variational operators during the evolution. The representation and the algorithms are validated in two problems in the area of system identification and evolutionary robotics.

Categories and Subject Descriptors: I.2.8 Heuristic methods

General Terms: Algorithms.

Keywords: Genetic algorithms, Recurrent fuzzy systems, Fuzzy control, Voronoi diagrams, Evolutionary Robotics

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 [8], while keeping an equivalence in knowledge rep-

resentation with other methods like neural networks and automata [6]. 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 [5]. 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. Most FLCs can be classified in two categories: the Mamdani type FLC where the output is computed as a combination of fuzzy numbers, and the Takagi-Sugeno (TS) type, where the output is defined as a linear combination of the inputs. In a TS type FLC with n inputs and m outputs, a typical rule has the following form:

$$\begin{aligned} R_i : \quad & \text{if } x_1 \text{ is } A_i^1 \text{ and } \dots \text{ and } x_n \text{ is } A_i^n \\ & \text{then } y_1 = a_{i0}^1 + \sum_j a_{ij}^1 x_j \\ & \dots \\ & y_m = a_{i0}^m + \sum_j a_{ij}^m x_j \end{aligned} \quad (1)$$

where $x_j (1 \leq j \leq n)$ are the input variables, $y_j (1 \leq j \leq m)$ are the output variables, $A_i^j (1 \leq j \leq n)$ are the fuzzy membership sets and $a_{ij}^k (0 \leq j \leq n, 1 \leq k \leq m)$ are the real valued parameters that define the linear approximation. The output of the complete FLC is computed by combining the outputs produced by all the rules, weighted by the degree of satisfaction of the antecedents.

Simple FLCs are usually defined by a trial and error process by using expert knowledge. However, automatic FLC generation methods are preferred for complex control systems [3] [21] [1]. Most FLC structures can be mapped into feed-forward neural networks, allowing the use of neural network learning algorithms to automate the design of FLC based on numerical data as well as on expert knowledge. The combined approach provides advantages from both worlds: the low level learning and computational power of neural networks is joined together with the high level human-like thinking and reasoning of fuzzy systems [17]. This combination has been very successful and there is a large number of models that combines fuzzy systems with neural networks [17], or even with standard PID control [23] [9].

The domain of application of these systems is limited to static problems due to its feed forward network structure [16]. 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. A very interesting approach that considers small order temporal problems with

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO'05, June 25–29, 2005, Washington, DC, USA.
Copyright 2005 ACM 1-59593-010-8/05/0006 ...\$5.00.

fuzzy logic is proposed in [2]. However, unless the number of delayed inputs and outputs is known before, it is not possible to define a feed forward model without backward connections that can process temporal sequences [10]. This is usually the case for most control problems, where this information is not known. However, recurrent structures can deal with this kind of problems. There is a large number of neural network models that have been proposed which are essentially feed forward structures with an extra set of units used to store previous activation values that are connected back to the inputs of other units.

By considering the large amount of recurrent neural network models that have been proposed, it is expected to see that most recurrent fuzzy systems are based on neural networks. For example, the RFNN (Recurrent Fuzzy Neural Network) model proposed in [16] defines recurrent connections in the second layer of the structure, which corresponds to the units that codifies the membership antecedent values. The RSONFIN (Recurrent Self Organizing Neural Fuzzy Inference Network) model proposed in [11] performs structure and parameter learning and includes an extra layer of units with recurrent connections that provides a kind of internal memory. The DFNN (Dynamic Fuzzy Neural Network) model proposed in [18] includes recurrent neural networks in the consequent in place of standard linear approximators like in the TS model. The TRFN (Takagi-Sugeno Type Recurrent Fuzzy Network) model proposed in [10] has an extra unit with recurrent connections for each fuzzy rule, which is responsible of memorizing the temporal history of activations of the rule. Other models like [12] and [24] follow similar approaches. In most cases, both supervised learning and non gradient based algorithms, like genetic algorithms, have been used to build or enhance the models.

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 this work, we propose a recurrent structure for fuzzy systems based on the Fuzzy Voronoi (FV) method proposed in [14], that allows the definition of recurrent fuzzy systems with a clear interpretation of recurrent units. The proposed Recurrent Fuzzy Voronoi (RFV) 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.

This paper is organized as follows. Section 2 describes the structure of the RFV model and the details on the geometrical basic structure. In section 3, the design of RFV models with evolutionary algorithms is analyzed. The properties of the proposed representation are discussed in section 4. In section 5, two experiments with the RFV model implemented with the EO library [15] are detailed. Finally, conclusions are presented in section 6.

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.

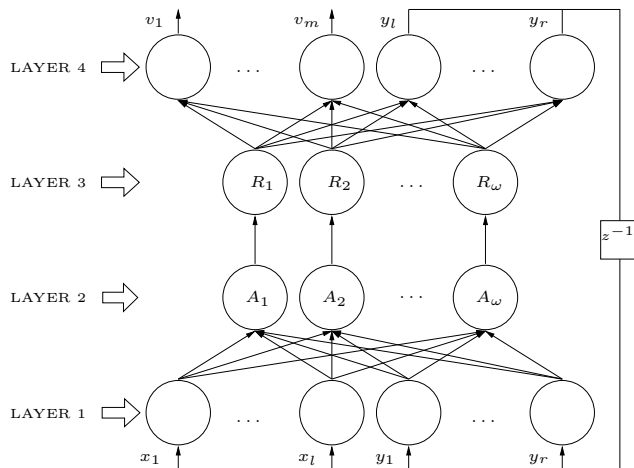


Figure 1: The structure of the RFV model

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.

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 2.3.

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:

$$O_i^k = (a_{k0}^i + \sum_j a_{kj}^i I_j) \mu_k(I) \quad (2)$$

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:

$$O_i = \sum_k O_i^k \quad (3)$$

2.2 Fuzzy reasoning model

The RFV model performs fuzzy inference by using rules defined as follows:

$$\begin{aligned} R_k : \quad & \mathbf{if} \quad I \text{ is } A_k \\ & \mathbf{then} \quad O_1 = a_{k0}^1 + \sum_j a_{kj}^1 I_j \\ & \quad \dots \\ & \quad O_{m+r} = a_{k0}^{m+r} + \sum_j a_{kj}^{m+r} I_j \end{aligned} \quad (4)$$

It can be noted that there is a strong similarity with the standard definition of TS rules given in equation 1. Except for the fact that now a single multidimensional set is used for membership, the main difference is that the fuzzy inference involves r terms in the input vector that are output values produced in the previous time step. The fuzzy system is a dynamic fuzzy inference system with the inferred values v_i produced in time $t + 1$ given by:

$$v_i(t + 1) = \sum_k O_i^k(t) \quad (5)$$

where the computation of the value $O_i^k(t)$ (see equation 2) involves the input vector $x(t)$ at time t and the internal values $y(t - 1)$ defined in time $t - 1$.

2.3 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 [4], a Voronoi diagram of a set of p points $\mathcal{P} = \{P_1, \dots, P_p\}$ 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 \mathcal{T} does not contain any point of \mathcal{P} in its interior. Figure 2 illustrates an example of a Voronoi diagram and 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 [4].

The FV representation [14] considers joint fuzzy sets defined from a Voronoi diagram $\mathcal{P} = \{P_1, \dots, P_p\}$. 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.

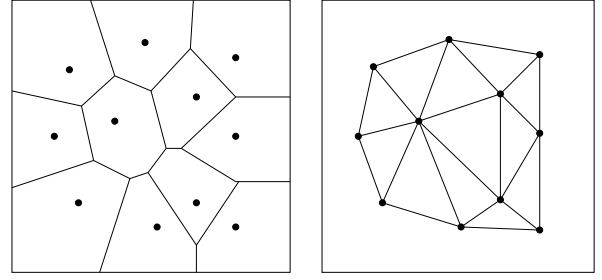


Figure 2: An example of a Voronoi diagram (left) and the corresponding Delaunay triangulation (right) for a set of points in \mathbb{R}^2

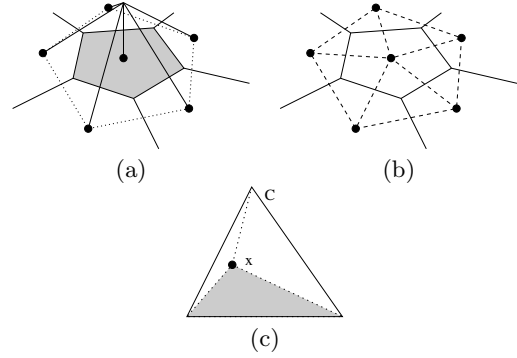


Figure 3: An example of a (a) joint fuzzy set for a single Voronoi region for $n = 2$, where the membership value is represented in the z-axis, and a (b) Voronoi diagram (solid line) and its corresponding Delaunay triangulation (dotted line) for $n = 2$. The graphic (c) shows an example of the membership 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

An example of such a joint fuzzy set is shown in figure 3-a for $n = 2$.

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

$$\mu_{S_k}(I) = \begin{cases} l_C(I) & x \in V_k \\ 0 & \text{elsewhere.} \end{cases} \quad (6)$$

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

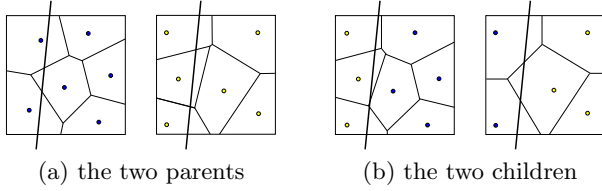


Figure 4: An example of the application of the Voronoi crossover in \mathbb{R}^2 . A random line is drawn of the input space partition of both parents (a), and the Voronoi regions (plus the associated rule parameters) are exchanged, leading to the two children (b).

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:

$$Ind = R_1 : \dots : R_\omega \quad (7)$$

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

$$R_i = P_i^1 : \dots : P_i^{l+r} : a_{i0}^1 : \dots : a_{i(l+r)}^{m+r} \quad (8)$$

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 [22, 13]. 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 (see section 4). Practical details on the algorithms, including all parameters, will be given in section 5. But before experimentally validating the FV representation, next section will discuss some of its properties.

4. PROPERTIES

First of all, the RFV representation belongs to the class of approximative representations [1], where each fuzzy rule defines its own fuzzy sets. It also provides continuous output, as most fuzzy systems. However, it also has a number of useful properties, that we shall now discuss in turn.

ϵ -completeness property: All RFV-based fuzzy systems defined with the RFV representation fulfills the completeness property at any required level, which establishes that any input must belong to at least one fuzzy set with a membership value not smaller than a threshold value ϵ :

$$\forall x \in U \exists A \in \{A_1, \dots, A_n\} \mu_A(x) \geq \epsilon. \quad (9)$$

For the RFV representation, it is clear from the definition of the membership function of equation (6) that this property will hold with $\epsilon = \frac{1}{2}$, as $l_C(x)$ will be above 0.5 if x lies in the Voronoi cell defined by C . This property guarantees an adequate representation for every input point, since there is always a rule that is applied with at least a known value of membership.

No need for genetic repair algorithms: Since it is not possible to define wrong or non complete fuzzy systems, the fuzzy systems produced by applying mutation or crossover operators are always valid control systems.

Adaptive fuzzy rules: The influence on the output of a particular fuzzy rule in the RFV representation does not only depend on the rule itself, it also depends on all neighbor rules. The area of application \mathcal{A}_k of a fuzzy rule R_k is defined as the union of all Delaunay regions which contain the point P_k , center of the rule R_k . Formally:

$$\mathcal{A}(R_k) = \bigcup_{P_k \in D_j} D_j \quad D_j \in D = \{D_1, \dots, D_\gamma\}. \quad (10)$$

where P_k is the center of the rule R_k and $D = \{D_1, \dots, D_\gamma\}$ is the Delaunay partition of the set $\mathcal{P} = \{P_1, \dots, P_p\}$. Figure 5 shows an example of the application area of some rules in a regular partition, and illustrates the interdependency of application areas of neighboring rules when some rules are removed or added. The evolutionary algorithm evolves individuals that represent complete fuzzy systems defined by a set of fuzzy rules that are synergistically related, and not fuzzy systems defined with a set of independent fuzzy rules. The variation operators hence modify the application areas of all fuzzy rules, while still maintaining the required ϵ -completeness level.

Adaptive a priori rules: 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 a priori 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 the synergistic effect described above, the evolutionary process, by adding rules more or less close to the a priori rules will also tune its domain of application – as will be clear on the experimental results in section 5.

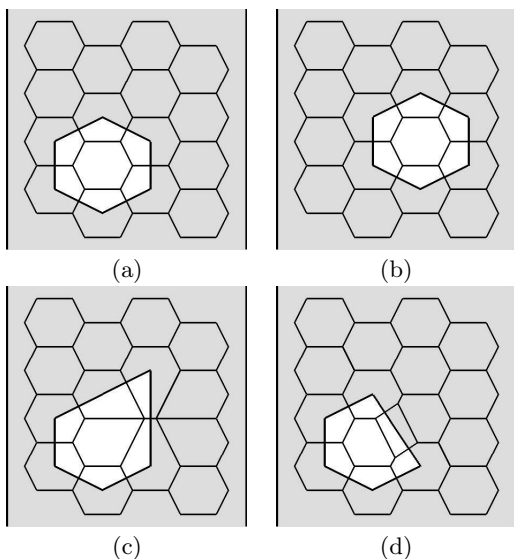


Figure 5: Diagram (a) shows the application area of a fuzzy rule and diagram (b) the application area of one of its neighbor rules. Diagram (c) shows the application area of the rule of diagram (a) when the rule of diagram (b) is removed, and diagram (d) the application area of the rule of diagram (a) when a rule is added between both rules.

Recurrent rules: The rules defined in the RFV controller are standard TS type fuzzy rules, with their own inputs and outputs. The complete system is recurrent because some outputs are connected to inputs, but each rule by itself is a standard TS type fuzzy rule. This fact contributes to provide a clear interpretation of the rules and make easy to define the a priori rules for the RFV controller. This approach contrasts with other models like RFNN [16], RSONFIN [11], DFNN [18] or the TRFN [10], where the rules themselves include backward connections. The recurrent connection model is similar to the NFSLS approach proposed in [20], except that in NFSLS only a single output is considered and standard fuzzy partition is performed in the input domain. Section 5 will introduce examples that show that this way of defining recurrent rules allows easy introduction of a priori knowledge.

5. EXPERIMENTS

In this section, the evolutionary approach to design RFV systems is evaluated in two problems. The first one is a system identification problem, where the outputs of the system are function of past inputs and outputs. This problem is introduced in order to compare the approach with other methods. The second problem is more interesting: it is an evolutionary robotic problem [21], where the ability to introduce a priori knowledge in the form of recursive rules is demonstrated.

5.1 System identification

The controlled plant is the same as used in the example 3 in [10] and is given by:

$$y_p(t+1) = \frac{y_p(t)y_p(t-1)(y_p(t)+2.5)}{1+y_p^2(t)+y_p^2(t-1)} + u(t) \quad (11)$$

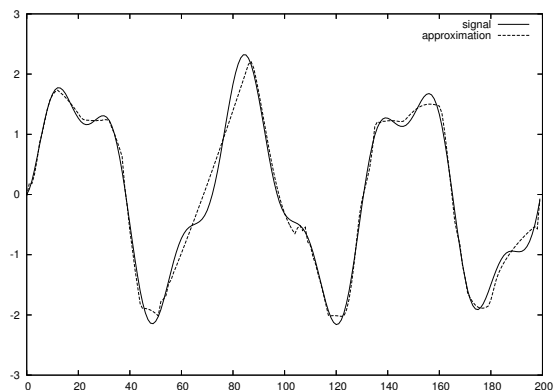


Figure 6: The test signal (dotted line) and the best approximation (continuous line) in the system identification problem.

where $y_p(t)$ and $u(t)$ are respectively the output and the input at time t . The desired output is defined by 250 pieces of data obtained from:

$$y_r(t+1) = 0.6y_r(t) + 0.2y_r(t-1) + 0.2\sin(2\pi t/25) + 0.4\sin(\pi t/32) \quad (12)$$

In the experiments, the population size is set to 50, the probability of Voronoi crossover is set to 0.8, Voronoi mutation to 0.3, mutation for addition and removal of Voronoi sites to 0.1, selection is performed by tournament with size 2, elitism is used and the number of generations is set to 1200. The fitness is defined as the RMS error. The test signal used and the best approximation found are shown in figure 6. The results for the best and average RMS error over 50 runs are listed in table 1. The table shows also the results obtained with the TRFN and RFNN models as presented in [10]. The comparison has to be considered with extreme care since, even if a careful selection of parameters was performed to replicate the experiments, there are some important differences in the models. For example, the RFV model uses variable length individuals, while the other methods use fixed length individuals. The main implication is that the number of fuzzy rules in the RFV model is determined by evolution, while in the other models has to be defined in advance. In the experiments with the TRFN and RFNN models performed in [10], the number of fuzzy rules is 4. The number of recurrent units is also 4, since in these models there is a one to one correspondence between the number of recurrent units and fuzzy rules. In the RFV model, the number of recurrent units (internal units) is not related with the number of rules, however it has to be defined in advance (see section 2) and cannot change during evolution. For the current experiment, the number of internal units is set to 1. Each evolutionary run took on average 374 seconds by using a Linux computer system with an Intel® Celeron® M D335 processor. Results in CPU time from [10] are not cited here since there is no information on the hardware used to run the experiments and the comparisons will not be fair.

5.2 Evolutionary robotics

A problem defined in the area of evolutionary robotics [21] has been selected to validate the RFV model. As a test base

Table 1: RMS error for the system identification experiment with one internal unit

RFNN+GA		TRFN+GA		RFV	
mean	best	mean	best	mean	best
0.3911	0.0850	0.0910	0.0536	0.0775	0.0235

for experiments, a simulated Khepera robot [19] 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 RFV controller is computed in a similar way as in [21], 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:

$$\text{fitness}(I) = \frac{1}{es} \sum_{i=1}^e \sum_{t=1}^s v(t) * (1 - a(t)) * (1 - d(t)) \quad (13)$$

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 [21] (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 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). In the experiments, the population size is set to 50, the probability of Voronoi crossover is set to 0.8, Voronoi mutation to 0.3, mutation for addition and removal of Voronoi sites to 0.1, selection is performed by tournament with size 2, elitism is used and

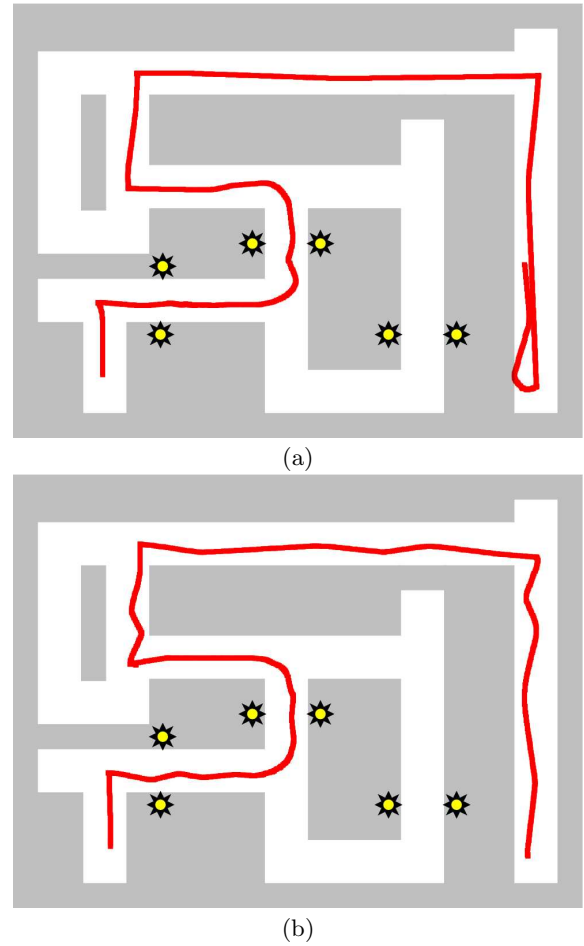


Figure 7: The performance of the best controller (a) without a priori knowledge and (b) with a priori knowledge when evaluated in a scenario not used during evolution. White areas correspond to corridors and stars represent lights. The starting position is on the bottom left side.

the 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.

The experiments were performed also by using a priori knowledge. The rules defined beforehand and inserted as explained in section 4 are shown in table 2.

The semantic of the 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

Rule	site	v_1			v_2			y_1		
		$a_0^1, a_1^1, a_2^1, a_3^1, a_4^1, a_5^1, a_6^1$	$a_0^2, a_1^2, a_2^2, a_3^2, a_4^2, a_5^2, a_6^2$	$a_0^3, a_1^3, a_2^3, a_3^3, a_4^3, a_5^3, a_6^3$	$a_0^4, a_1^4, a_2^4, a_3^4, a_4^4, a_5^4, a_6^4$	$a_0^5, a_1^5, a_2^5, a_3^5, a_4^5, a_5^5, a_6^5$	$a_0^6, a_1^6, a_2^6, a_3^6, a_4^6, a_5^6, a_6^6$			
R_1	0,0,0,0,0,0	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	0,0,0,0,0,0,0	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	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	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	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	1,0,0,0,0,0,0	0,0,0,0,0,0,0	0,0,0,0,0,0,0	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	1,0,0,0,0,0,0	1,0,0,0,0,0,0	0,0,0,0,0,0,0	0,0,0,0,0,0,0			

Table 2: A Priori 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.

Table 3: Fitness for the evolutionary robot experiment

RFV			RFV + a priori knowledge		
mean	best	var	mean	best	var
0.7728	0.8723	0.0686	0.8510	0.8835	0.0255

y_1 is reset to 0. It is important to note that the a priori knowledge is 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, as it was detailed in section 4.

The results for the best and average fitness over 10 runs are listed in table 3, where the best possible value for fitness is 1 and 0 is the worst. Faster learning and smaller error are achieved with a priori rules, but also the standard deviation is smaller, meaning that it is a more robust approach. Figure 7 shows the performance of the best controllers found during evolution in a scenario not used during evolution. The controllers are evaluated for 800 time steps. The controller that does not use a priori knowledge can drive the robot for a longer distance in the same number of time steps, performing not so abrupt turns in the intersections but both controllers can drive the robots by following the light signals as expected.

Table 4 shows two rules obtained through evolution in a selected experiment. The first rule is applied when there are obstacles on the right and front sides of the robot, with high light conditions (0.86) and the light signal (internal unit) keeps a medium light value (0.48). The output produces (in the center point of the region) a turn to the left and higher value in the light signal unit (0.83). The second rule applies when there are obstacles on the back and on the right side, with low light conditions (0) and with the light signal with a medium value (0.63) and produces a turn to the left and a higher value in the light signal (0.88).

However, the most important point is that a definite semantic interpretation of the internal unit is provided with the a priori 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 a priori knowledge. Figure 8 shows the value of the internal unit of both best controllers plotted for the 800 time steps when evaluated on the test scenario from figure 7.

The value of the internal unit for the controller evolved with a priori knowledge represents the expected semantics, with two peaks on the areas where light signals were de-

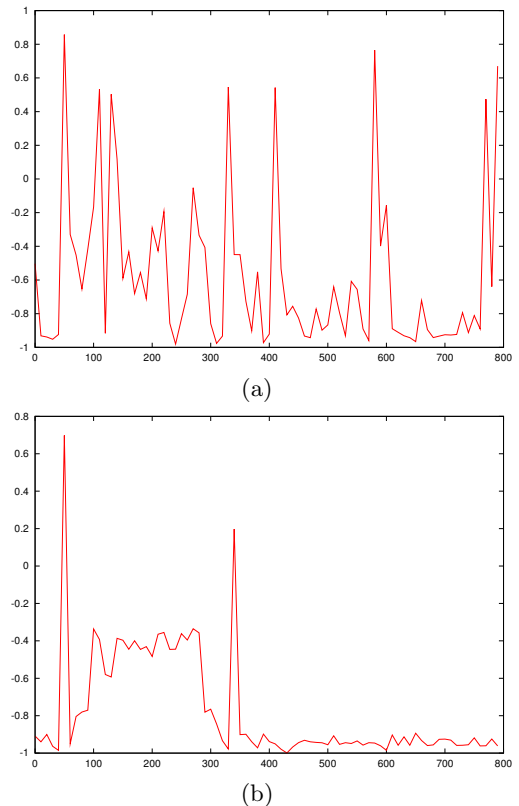


Figure 8: Value of the internal unit of the best controller obtained through evolution (a) without and (b) with a priori knowledge when evaluated on the test scenario of figure 7.

tected. 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.

6. CONCLUSIONS

In this paper, the RFV model has been proposed. This model is an extension of the FV model defined to extend the application domain to include temporal problems. The temporal relations are embedded by including internal units that provide feedback by connecting outputs to inputs. These internal units act as memory elements. This paper propose the use of evolutionary algorithms as a design tool of the RFV

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	0.4,1,0.8,0,0.9,0.5	0.9,0.9,0.2,0.9,0.4,0.3	0.4,0.9,0.9,0.8,1,0.5,0.7	0.1,0.7,0.5,0.8,0.6,1,1
R_2	0.4,0.5,0.7,0.9,0,0.6	0.9,0.3,0.7,0.3,0.4,0.7,0.8	0.9,0.5,0.6,0.9,0.7,0.2,0.2	0.2,0.2,0.8,1,0.1,0,0

Table 4: An example of two rules obtained through evolution (rounded values).

model. The controllers are represented by following the FV model, which is a representation for fuzzy controllers based on Voronoi diagrams that can represent fuzzy systems with synergistic rules, fulfilling the ϵ -completeness property and providing a simple way to introduce a priori knowledge. The geometric interpretation of the rules allows the use of geometric variational operators that proved to be useful also in other contexts. The main benefit of the proposed representation is the possibility to provide a definite semantic to the internal (or recurrent) units. The representation and the algorithms have been validated in two problems in the area of system identification and evolutionary robotics. Future work include experiments on a real mobile robot, the study of the impact of using the so-called *Symbolic Controllers* approach [7], and an analysis of problems with higher number of variables with and without expert knowledge.

7. 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] P. Carinena, C. Regueiro, A. Otero, A. Bugarin, and S. Barro. Landmark detection in mobile robotics using fuzzy temporal rules. *IEEE Transactions on Fuzzy Systems*, 12(4):423–435, August 2004.
- [3] O. Cerdón and F. Herrera. A two-stage evolutionary process for designing task fuzzy rule-based systems. *IEEE Transactions on Systems, Man and Cybernetics*, 29(6):703–714, 1999.
- [4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications*. Springer Verlag, 1998.
- [5] G. Feng. An approach to adaptive control of fuzzy dynamic systems. *IEEE Transactions on Fuzzy Systems*, 10(2):268–275, April 2002.
- [6] C. L. Giles, C. W. Omlin, and K. K. Thornber. Equivalence in knowledge representation: Automata, recurrent neural networks and dynamical fuzzy systems. *Proc. of the IEEE*, 87(9):1623–1640, September 1999.
- [7] N. Godzik, M. Schoenauer, and M. Sebag. Evolving symbolic controllers. In G. R. et al., editor, *Applications of Evolutionary Computing*, Lecture Notes in Computer Science 2611, pages 638–650, 2003.
- [8] H. A. Hagras. A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Transactions on Fuzzy Systems*, 12(4):524–539, 2004.
- [9] M. Hojati and S. Gazor. Hybrid adaptive fuzzy identification and control of nonlinear systems. *IEEE Transactions on Fuzzy Systems*, 10(2):211–221, 2002.
- [10] C.-F. Juang. A task-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 10(2):155–170, April 2002.
- [11] C.-F. Juang and C.-T. Lin. A recurrent self-organizing neural fuzzy inference network. *IEEE Transactions on Neural Networks*, 10(4):828–845, July 1999.
- [12] N. Kasabov and Q. Song. Denfis: Dynamic evolving neural fuzzy inference systems and its application for time series prediction. *IEEE Transactions on Fuzzy Systems*, 10(2):144–154, April 2002.
- [13] C. Kavka and M. Schoenauer. Voronoi diagrams based function identification. *GECCO, Lecture Notes in Computer Science*, 2723:1089–1100, 2003.
- [14] C. Kavka and M. Schoenauer. Evolution of voronoi based fuzzy controllers. *PPSN, Lecture Notes in Computer Science*, 3242:541–550, 2004.
- [15] M. Keijzer, J. Merelo, G. Romero, and M. Schoenauer. Evolving objects: A general purpose evolutionary computation library. In P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, editors, *Proceedings of the Fifth Conference on Artificial Evolution (AE-2001)*, volume 2310, pages 231–244. Springer Verlag, 2001.
- [16] C.-H. Lee and C.-C. Teng. Identification and control of dynamic systems using recurrent fuzzy neural networks. *IEEE Transactions on Fuzzy Systems*, 8(4):349–366, August 2000.
- [17] C. T. Lin and S. G. Lee. *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*. Prentice Hall, Englewood Cliffs, NJ, 1986.
- [18] P. A. Mastorocostas and J. B. Theocharis. A recurrent fuzzy-neural model for dynamic system identification. *IEEE Transactions on Systems, Man and Cybernetics*, 32(2):176–190, April 2002.
- [19] O. Michel. Khepera simulator package version 2.0: Freeware mobile robot simulator. <http://wwwi3s.unice.fr/om/khep-sim.html>.
- [20] G. Mouzouris and J. Mendel. Dynamic non-singleton fuzzy logic systems for nonlinear modeling. *IEEE Transactions on Fuzzy Systems*, 5(2):199–208, 1997.
- [21] S. Nolfi and D. Floreano. *Evolutionary Robotics, The Biology, Intelligence, and Technology of Self-Organizing Machines*. Bradford Books, 2000.
- [22] 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.
- [23] Y. L. Sun and M. J. Er. Hybrid fuzzy control of robotic systems. *IEEE Transactions on Fuzzy Systems*, 12(6):755–765, December 2004.
- [24] J. Zhang and J. Morris. Recurrent neuro fuzzy models for nonlinear process modeling. *IEEE Transactions on Neural Networks*, 10(2):313–326, March 1999.