Modified Reinforcement Learning-Hierarchical Neuro-Fuzzy Politree Model for Control of Autonomous Agents

KARLA FIGUEIREDO♦, MARCIO SANTOS♣, MARLEY VELLASCO♣ & MARCO PACHECO♣

♦*Electronics & Telecommunication Engineering, UERJ, Brazil*
karlaf@uerj.br

♣*Electrical Engineering, PUC-Rio – Brazil*
bs.in@uol.com.br
marley@ele.puc-rio.br
marco@ele.puc-rio.br

Abstract: This work presents a new hybrid neuro-fuzzy model for automatic learning of actions taken by agents. The main objective of this new model is to provide an agent with intelligence, making it capable, by interacting with its environment, to acquire and retain knowledge for reasoning (infer an action). This new model, named Reinforcement Learning Hierarchical Neuro-Fuzzy Politree (RL-HNFP), and its improved version (RL-HNFP$^+$), descends from the Reinforcement Learning Hierarchical Neuro-Fuzzy BSP (RL-HNFB) that applies binary hierarchical partitioning methods, together with the Reinforcement Learning (RL) methodology. These two characteristics permit the autonomous agent to automatically learn its structure and its necessary action in each position in the environment. The RL-HNFP model and its extension (RL-HNFP$^+$) are evaluated in a well known benchmark application in the area of autonomous agents: the Mountain Car Problem. The results obtained demonstrate the potential of these models, which operate without any prior information, such as number of rules, rules specification, or number of partitions that the input space should have, providing good performance and demonstrating the agent's autonomy.

Keywords: *Neuro-Fuzzy Systems; Learning; Reinforcement Learning, Automatic Learning*

1. INTRODUCTION

This work presents the hybrid *Reinforcement Learning Hierarchical Neuro-Fuzzy Politree* model (RL-HNFP) [Figueiredo et al., 2004], [Figueiredo et al., 2005] and its new improved version (RL-HNFP$^+$), which accomplishes the learning process through Reinforcement Learning algorithm.

Hybrid Intelligent Systems conceived with the use of techniques such as Fuzzy Logic (FL) [Klir and Yuan, 1995], [Mendel, 1995] and Neural Networks (NN) [Bishop, 1995], [Haykin, 1998] have been applied in areas where traditional approaches were unable to provide satisfactory solutions. Many researchers have attempted to integrate these two techniques by generating hybrid models that associate their advantages and minimize their limitations and deficiencies. With this objective in mind, hybrid neuro-fuzzy systems, or simply, neuro-fuzzy systems [Jang and Sun, 1997], [Lin and Lee, 1996] have been created. These systems combine the learning capacity of neural networks with the linguistic interpretability of fuzzy inference systems. Traditional neuro-fuzzy models [Abraham, 2005], such as ANFIS [Jang, 1993], NEFCLASS [Kruse and Nauck, 1995] and FSOM [Vuorimaa, 1994], have a limited capacity for creating their own structure and rules [Souza et al., 2002a]. In addition, most of these models employ grid partition of the input space, which, due to the rule explosion problem, makes them more adequate for applications with a smaller number of inputs. When a greater number of input variables are necessary, the system's performance deteriorates.

Finally, most of these traditional neuro-fuzzy systems depend on supervised training, where a desired output must be provided for each input pattern. When the information about the desired output is not available (usually the case in autonomous agent applications), the learning procedure must be carried out by means of a *Reinforcement Learning* algorithm. Furthermore, when the environment is large and/or continuous, the application of traditional Reinforcement Learning (RL) methods based on lookup tables (a table that stores value functions for a small or discrete state space) is no longer feasible because the state space becomes too large. This problem is known as the *curse of dimensionality* [Sutton, 1996], [Boyan and Moore, 1995], [Jouffe, 1998]. In order

to bypass it, some form of generalization must be incorporated into the representation of states.
In general, in addition to their limitation regarding the number of input variables, the main RL- and Fuzzy Logic-based models require predefinitions such as: number and format of the fuzzy sets used in the fuzzy rules' antecedents and consequents; number of rules; and even the specification of the antecedents that compose the rules.

Thus, the main motivation of this work was to devise neuro-fuzzy models that could overcome these basic limitations, so that an autonomous agent could learn the actions that must be carried out in large and/or continuous environments, without prior knowledge of the environment's dynamics and with the least possible prior definitions.

The models described in this paper were, therefore, developed based on an analysis of the limitations in the existing models and of the desirable characteristics for RL-based learning systems, particularly in applications involving continuous and/or high dimensional environments [Sutton, 1996], [Boyan and Moore, 1995], [Jouffe, 1998], [Moore, 1991]. Thus, a recursive space partitioning methodology (which had already been explored with excellent results in previous works [Jang and Sun, 1997], [Gonçalves et al., 2004], [Vellasco et al., 2004] and [Souza et al., 2002b]) was chosen, which significantly reduces the limitations of the existing neuro-fuzzy systems. The use of recursive partitioning, combined with *Reinforcement Learning,* resulted in a new class of Neuro-Fuzzy Systems called *Reinforcement Learning-Hierarchical Neuro-Fuzzy Politree System* (RL-HNFP) [Figueiredo et al., 2004] and [Figueiredo et al., 2005].

The functionality of the model's fuzzy component is to aggregate states that have similar behaviours and associate them with one and the same action. The RL component makes the model able to learn the most suitable action to be executed for a given state. The hierarchical aspect is related to the fact that each partition of the input space defines a subsystem which, if necessary, may have a subsystem with the same structure as its consequent (recursiveness). This means that the hierarchical tree structure increases according to the complexity of the problem. This characteristic smoothens the value function generalization process (with the hierarchical fuzzy tree as the function approximator) without affecting the results, which is a good requirement for process convergence [Sutton, 1996].
In this manner, the RL-HNFP model presents the following important characteristics: it automatically learns the model's structure; it performs self-adjustment of the parameters associated with the structure; it is capable of learning an action to be taken when the agent is in a given state of the environment; it is able to deal with a greater number of inputs when compared with traditional neuro-fuzzy systems; and it automatically generates linguistic rules. These characteristics represent an important differential in relation to the existing intelligent agent learning systems.

However, the original RL-HNFP presents a drawback related to the defuzzification method, which caused longer training process. In the original model, rules associated with low activated partitions were considered in the defuzzification process, which deteriorated the model's output.

Therefore, this paper also presents an extended version of RL-HNFP model, where the resultant output value (y) depends only on the most activated rules, which improves the model's performance.

This paper has been organized in four additional sections. Section 2 contains a brief description of the Politree partitioning. Section 3 introduces the RL-HNFP model, describing its basic cell, its architecture and its learning method. Section 4 presents the results obtained with the *mountain-car problem* [Sutton, 1996], a benchmark of the area. The results obtained by the RL-HNFP model and its extension are compared with models developed by other researchers. Lastly, section 5 presents the conclusions.

2. POLITREE PARTITIONING

The partitioning process of the input/output space has great influence on the neuro-fuzzy system performance in relation to its desirable features (accuracy, generalization, automatic generation of rules, etc). This work proposes the use of recursive partitioning methods of the input/output space, resulting in a new class of neuro-fuzzy system that overcomes two of the main weaknesses of current Neuro-Fuzzy Systems: their fixed structure and limited number of inputs.

The Politree partitioning was inspired by the quadtree structure proposed by Finkel and Bentley [Finkel and Bentley, 1974], which has been widely used in the area of images manipulation and compression. In this partitioning, the n-dimensional space is successively subdivided in quadrants that, in turn, can be again subdivided in four regions (quadrants) in a recursive operation. The limitation of the Quadtree partitioning (fixed or adaptive) is in the fact that it works only in two-dimensional spaces. The Politree partitioning, used in the Reinforcement Learning - Hierarchical Neuro-Fuzzy Politree model (RL-HNFP), is a generalization of this idea. In this partitioning the subdivision of the

n-dimensional space is accomplished in $m=2^n$ subdivisions.

The Politree partitioning can be represented by a tree structure. Figure 1a shows the partitioning for the particular case when n = 2 (two inputs) and Figure 1b presents the generic politree partitioning (with n inputs).

Politree partitioning is flexible and minimizes the exponential rule growth problem, since it creates new rules locally according to the learning process. This type of partitioning is considered recursive because it makes use of a recursive process to generate partitions. In this manner, the resulting models have a hierarchy in their structure and consequently, hierarchical rules.

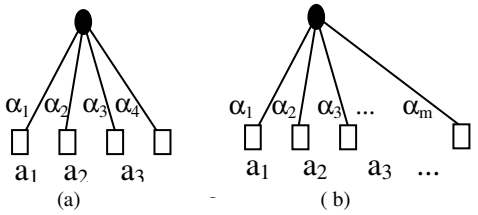


Figure 1. (a) tree representation of a Politree structure with two inputs; (b) generic tree representation of the politree partitioning with n inputs, where $m = 2^n$.

3. REINFORCEMENT LEARNING – HIERARCHICAL NEURO-FUZZY POLITREE (RL_HNFP)

The RL-HNFP model is made up of one or several standard cells called RL-Neuro-Fuzzy politree (RL-NFP) cells. These cells are arranged in a hierarchical structure in the form of a tree. The outputs of the cells in the lower levels are the consequents of the cells in the higher levels. The following sub-sections describe the basic cell, the hierarchical structure and the learning algorithm.

3.1. Original RL Neuro-Fuzzy Politree Cell

An RL-NFP cell is a mini-neuro-fuzzy system that performs politree partitioning in a given space according to the membership functions described in equation (1), where ρ(x) and μ(x) represent the low and high membership functions, respectively. The RL-NFP cell generates a precise (crisp) output after the defuzzification process. The values of the input variables are read by the agent's *sensors* and then are inferred in the antecedents' fuzzy sets (*low* and *high*).

$$\mu(x) = \frac{1}{1+e^{-(a(x-b))}} \quad and \quad \rho(x) = 1-\mu(x) \qquad (1)$$

Each cell receives all the inputs that are being considered in the problem. For illustration purpose, Figure 2 depicts a cell with two inputs – x_1 and x_2 - (Quadtree partitioning), providing a simpler representation than the n-dimensional form proposed by Politree. In Figure 2, each partitioning is generated by the combination of the two membership functions ρ (*low*) and μ (*high*) of each input variable and is associated with a set of actions (a_1, a_2,..., a_t). The simplified representation of the basic cell is presented in Figure 3.

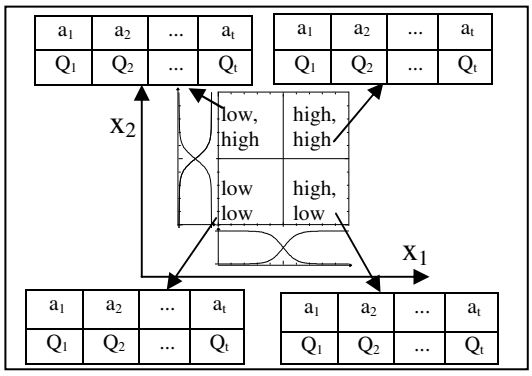


Figure 2. Internal representation of the RL-Neuro-Fuzzy Politree cell with two inputs.

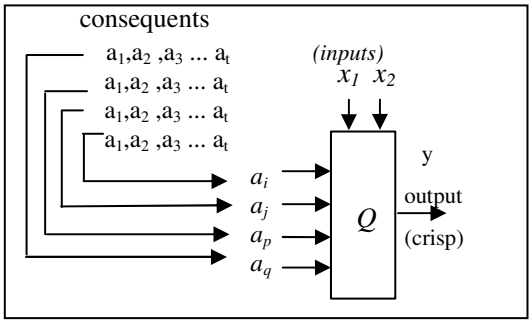


Figure 3. RL-NFP cell schematic.

The consequents of the cell's low and high poli-partitions may be of the *singleton* type or the output of a stage of a previous level. Although the *singleton* consequent is simple, this consequent is not previously known because each *singleton* consequent is associated with an action that has not been defined a priori. Each poli-partition has a set of possible actions (a_1, a_2, ... a_t), as shown in Figure 2, and each action is associated with a *Q*-value function. The *Q*-value is defined as being the sum of the expected values of the rewards obtained by the execution of action *a* in state *s*, in accordance with a policy π. For further details, see [Sutton and Barto, 1998].

By means of the RL-based learning algorithm (see section 3.3), one action of each poli-partition (for

example, a_i, a_j, a_p and a_q) is selected (Figure 3) as the one that represents the desired behavior of the system whenever that system is in a given state. Thus, the consequents are the actions that the agent must learn along the process.
The linguistic interpretation of the mapping implemented by the RL-NFP cell shown in Figure 3 is given by the following set of rules:

rule$_1$: If $x_1 \in \rho_1$ and $x_2 \in \rho_2$ then $y = a_i$
rule$_2$: If $x_1 \in \rho_1$ and $x_2 \in \mu_2$ then $y = a_j$
rule$_3$: If $x_1 \in \mu_1$ and $x_2 \in \rho_2$ then $y = a_p$
rule$_4$: If $x_1 \in \mu_1$ and $x_2 \in \mu_2$ then $y = a_q$

Each rule corresponds to one of the four poli-partitions generated by the Politree partitioning. When the inputs fall in quadrant (*low*, *low*), rule 1 has the greatest firing level. Each quadrant, in turn, can be subdivided in four parts, through another RL-NFP cell.

The most suitable defuzzification method in this case is the weighted average because it combines the consequents of the fuzzy rules with each of these rules' firing level, thereby generating an output (crisp) '*y*' according to the expression given in equation (2), where α_i is the firing level of rule *I* (poli-partition *i*) and consequent a_i corresponds to one of the two possible consequents below:
a singleton (fuzzy singleton consequent, or zero-order Sugeno): the case where a_i = constant;
the output of a stage of a previous level: the case where $a_i = y_m$, where y_m represents the output of a generic cell 'm', whose value is also calculated by equation (2).

$$y = \left(\frac{\sum_{i=1}^{2^n} \alpha_i \times a_i}{\sum_{i=1}^{2^n} \alpha_i} \right) \qquad (2)$$

$$y = \sum_{i=1}^{2^n} \alpha_i . a_i \qquad (3)$$

Since the *high* (μ) and *low* (ρ) membership functions are complementary, the defuzzification process is simplified (eq. 3), with the denominator of equation (2) being equal to 1 for any values of inputs '*x*'.

3.2. RL-HNFP Architecture

Once the system's basic cell has been described, RL-HNFP models can be created based on the interconnection of these cells. The RL-NFP cells form a hierarchical structure that results in the rules that compose the agent's reasoning. Figure 4 exemplifies this architecture.

In the architecture presented in Figure 4, the poli-partitions 1, 3, 4, … m-1 have not been subdivided, having as consequents of their rules the values a_1, a_3, a_4, … a_{m-1}, respectively. On the other hand, poli-partitions 2 and m have been subdivided; so the consequents of their rules are the outputs (y_2 and y_m) of subsystems 2 and m, respectively.

On its turn, these subsystems have, as consequent, the values a_{21}, a_{22}, ..., a_{2m}, and a_{m1}, a_{m2}, ..., a_{mm}, respectively. Each 'a_i' corresponds to a consequent of zero-order Sugeno (singleton), representing the action that will be identified (between the possible actions), through reinforcement learning, as being the most favorable for a certain state of the environment.

The output of the system depicted in Figure 4 (defuzzification) is given by equation (4). Again, in this equation, α_i corresponds to the firing level of partition *i* and a_i is the singleton consequent of the rule associated with partition *i*. Equation (4) considers that the firing level of the inactive cells have already been distributed, as will be explained in Section 3.2.1.

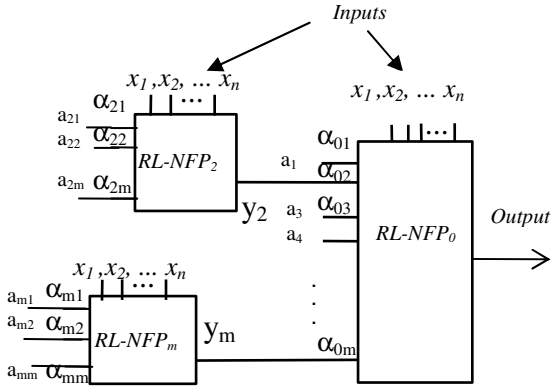


Figure 4. Example of architecture of the RL-HNFP model

$$y = \alpha_1 . a_1 + \alpha_2 \sum_{i=1}^{2^n} \alpha_{2i} . a_{2i} + \alpha_3 . a_3 + \alpha_4 . a_4 + \\ + \ldots + \alpha_m \sum_{i=1}^{2^n} \alpha_{mi} . a_{mi} \qquad (4)$$

The structure's output *y* is the resultant action (crisp value) that the agent's *actuator* must perform. Therefore, the RL-HNFP model creates and determines its structure by mapping states into actions.

The hierarchical characteristic of the RL-HNFP model smoothens the value function generalization process (with the hierarchical fuzzy tree as the function approximator) without affecting the results,

which is a good requirement for process convergence [Sutton, 1996].

3.2.1. Modified RL Neuro-Fuzzy Politree Cell

The original RL-HNFP defuzzification process, presented in section 3.1, considered all activated partitions, including those with low activation value, which jeopardized the learning process causing longer training cycles.

To understand the problem with the original model, consider the HNFP architecture presented in Figure 4. When the RL-NFP_2 cell is not active (its inputs' values are out of the domain associated to this cell, resulting in $\alpha_{21}, \alpha_{22}, \ldots \alpha_{2m} = 0$), the output value of this cell (y_2) is equal to zero. Therefore, in spite of α_{02} (RL-NFP_0) being non zero, the resultant product of $\alpha_{02} * y_2$ is zero. In the defuzzification process of cell RL-NFP_0, however, α_{02} is still considered in the denominator of the defuzzification formula, which always sums 1, as shown in equation (5).

$$y = \left(\frac{\alpha_{01} \times a_1 + \alpha_{02} \times y_2 + \ldots + \alpha_{0m} \times y_m}{\alpha_{01} + \alpha_{02} + \ldots + \alpha_{0m}} \right) \qquad (5)$$

Therefore, the defuzzification process was altered, generating the improved RL-$HNFP^+$ model. In this extended model the system's output (y) depends just on the most activated rules. Figure 4 also illustrates this modification with the three cells architecture: RL-NFP_0, RL-NFP_2 and RL-NFP_m, each with n inputs.

As shown in Figure 4, RL-NFP_2 and RL-NFP_m cells descend from RL-NFP_0 cell. When the RL-NFP_2 cell is not active, the firing level $\alpha_{21}, \alpha_{22}, \alpha_{23}$ and α_{2m} are all zero and, therefore, the firing level (α_{02}) can be divided and distributed to the firing level of the others partitions (α_{01}, $\alpha_{03} \ldots \alpha_{0m}$).

This division is in accordance with the contribution of each firing level of rule ($\alpha_{01} / \sum \alpha_{01} + \alpha_{03} + \ldots + \alpha_{0m}$, $\alpha_{03} / \sum \alpha_{01} + \alpha_{03} + \ldots + \alpha_{0m}$ $\cdots \alpha_{0m} / \sum \alpha_{01} + \alpha_{03} \ldots + \alpha_{0m}$). This way, the structure's output y depends only on the active rules. That is equivalent to subtract α_{02} of the denominator of equation 2.

3.3. RL_HNFP Learning Algorithm

Neuro-fuzzy learning is generally divided into two parts: structure identification and parameter adjustment. The RL-HNFP model performs both of these learning tasks in a single algorithm.

In the RL-HNFP model the reinforcement learning process is based on the SARSA [Sutton and Barto, 1998] method. The state identification process, which is not previously known, is accomplished by the learning phase.

The flowchart shown in Figure 5 describes the learning algorithm of the RL-HNFP model.

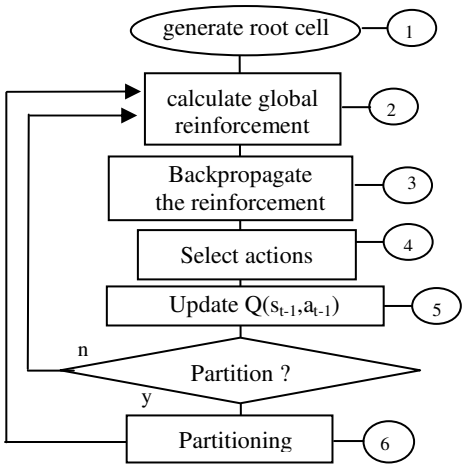


Figure 5. Learning Algorithm of RL-HNFP model

The agent must run many cycles in order to ensure that the system/environment in which it has been inserted is learned. A cycle is defined as the number of steps the agent takes in the environment from the point at which that agent has been initialized to the point considered as being its goal. Each step comprises the complete execution of the algorithm, from the agent's reading the environment to the execution of the action.

The learning process starts out with the definition of the input variables that are relevant to the system/environment in which an agent is inserted and of the set of actions it may use in order to attain its objectives.

Step 1 - Generating a father-cell (or root-cell)
A root-cell is created with fuzzy sets whose domain is equal to the universe of discourse of the cell's input variables x_i. The values of the input variables are read by the agent's *sensors,* are normalized, and each x_i value is evaluated in the high and low fuzzy sets, resulting in two membership degrees, $\rho(x_i)$ and $\mu(x_i)$, respectively. Each poli-partition chooses one of the actions (from its set of actions), based on the methods described in step 4 of this algorithm. The cell output is calculated by the defuzzification process given by equation (3) (considering the modification described in section 3.2.1) and represents the action that will be executed by the agents' actuators.

Step 2 - Global Reward:
After the action is carried out, the environment is read once again. This reading enables the calculation

of the environment global reward value that will be used to evaluate the action taken by the agent. This value must be calculated by means of an evaluation function defined according to the agent's objectives. Hence, this evaluation function defines not only the nature of the reinforcement, by rewarding or punishing (+1/-1), but also its intensity, thereby making for a more efficient process of guiding this agent during the learning.

Step 3 - Backpropagation of the Reinforcement:
At each step, the reinforcement (reward) is calculated for each partition of all active cells by means of its participation in the resulting action. Thus, the environment reinforcement calculated by the evaluation function is backpropagated from the root-cell to the leaf-cells, according to Figure 6 and to the calculations shown in equations (6) to (9).

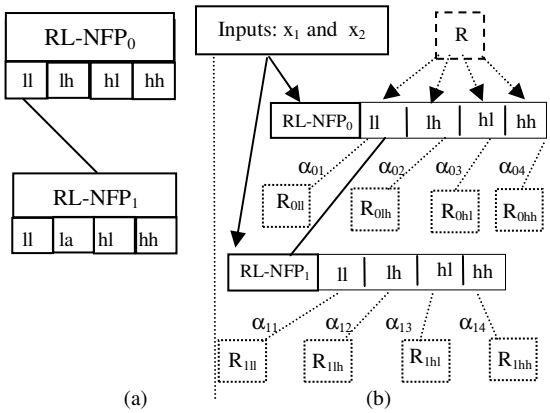


Figure 6. Backpropagation of the global reinforcement of the environment, where the cell root (RL-NFP$_0$) has one descendant in the partition low/low (RL-NFP$_1$). In this figure *ll*, *lh*, *hl* and *hh* indicate low-low, low-high, high-low and high-high partitions, respectively.

$$\begin{aligned} \alpha_{01} &= \rho_0(x_1) \,.\, \rho_0(x_2) \\ \alpha_{02} &= \rho_0(x_1) \,.\, \mu_0(x_2) \\ \alpha_{03} &= \mu_0(x_1) \,.\, \rho_0(x_2) \\ \alpha_{04} &= \mu_0(x_1) \,.\, \mu_0(x_2) \end{aligned} \quad (6)$$

$$\begin{aligned} \alpha_{11} &= \rho_1(x_1) \,.\, \rho_1(x_2) \\ \alpha_{12} &= \rho_1(x_1) \,.\, \mu_1(x_2) \\ \alpha_{13} &= \mu_1(x_1) \,.\, \rho_1(x_2) \\ \alpha_{14} &= \mu_1(x_1) \,.\, \mu_1(x_2) \end{aligned} \quad (7)$$

$$\begin{aligned} R_{0ll} &= \alpha_{01} \,.\, R_{global} \\ R_{0lh} &= \alpha_{02} \,.\, R_{global} \\ R_{0hl} &= \alpha_{03} \,.\, R_{global} \\ R_{0hh} &= \alpha_{04} \,.\, R_{global} \end{aligned} \quad (8)$$

$$\begin{aligned} R_{1ll} &= \alpha_{11} \,.\, R_{0ll} \\ R_{1lh} &= \alpha_{12} \,.\, R_{0ll} \\ R_{1hl} &= \alpha_{13} \,.\, R_{0ll} \\ R_{1hh} &= \alpha_{14} \,.\, R_{0ll} \end{aligned} \quad (9)$$

The values α_i are calculated using the T-norm AND operator. Equations (6) and (7) show the calculations of the α_i for the root cell (RL-NFP$_0$) and its descendant (RL-NFP1), respectively. The calculations of the reinforcements for cells RL-NFP$_0$ and RL-NFP$_1$ are defined by eqs. (8) and (9), respectively (R_{global} is the calculated global reinforcement).

Step 4 - Selection of the actions:
The actions are associated to Q-value functions and compose a set of actions that are selected and experimented during the RL learning. Exploring the space of states is a key issue to discover actions that correspond to the best agent's response (with regards to achieving an objective) when the agent is in a given state of the environment. So, the ϵ-*greedy* [Sutton and Barto, 1998] method was used, which selects an action associated to the highest expected Q-value with (1 - ϵ) probability and with ϵ probability it randomly selects any action. The maximum value of ϵ is 0.1 (10%) [Sutton and Barto, 1998].

Step 5 - Update Q(s_{t-1}, a_{t-1}):
After the calculation of the reinforcement values for each cell in the structure, the Q-values associated to actions that have contributed to the resulting action must be updated.

This update is based on the evaluation between the current and previous global returns. The Q-values update takes place in two distinct forms: when the current global reinforcement value is higher than the previous global reinforcement ($R_{t+1} > R_t$) the objective is to reward by increasing the Q value, following a similar formula to SARSA method [Sutton and Barto, 1998]. In this case ($R_{t+1} > R_t$), there is a reduction in the ϵ-greedy exploration/exploitation rates that are associated with each bi-partition (set of actions) of each active cell. On the other hand, when the current global reinforcement is lower than or equal to the previous global reinforcement ($R_t \geq R_{t+1}$), the objective is to punish by reducing the Q value, so as to reduce the probability of this action being chosen when the cell is active again [Figueiredo et al., 2004],[Figueiredo et al., 2005]. The rates of the ϵ-greedy parameters of the partitions involved are then increased, which allows different actions to have more chances of being chosen when this bi-partition is active again.

Step 6 - Partitioning:
In order for a cell to be partitioned, each poli-partition must satisfy two criteria. The first criterion prevents the structure from growing as a result of a bad performance caused by a still immature choice of actions; the second encourages partitioning when there are significant variations in the actions' value functions. When a poli-partition has all the

necessary requirements for partitioning, a leaf cell is created and connected to that poli-partition. Its domain will be the sub-domain that corresponds to the poli-partition ancestor. Leaf cells also inherit the set of actions with its respective Q-values from their ancestor.

The original RL_HNFP and its extended version RL-HNFP$^+$ models have been evaluated in a common benchmark application. The case study is presented in the next section.

4. STUDIES

4.1. Mountain Car Problem

The mountain car (see Figure 7) is a highly relevant benchmark that has been used by different researchers [Sutton, 1996], [Boyan and Moore, 1995], [Jouffe, 1998] with the purpose of testing their function approach methods.

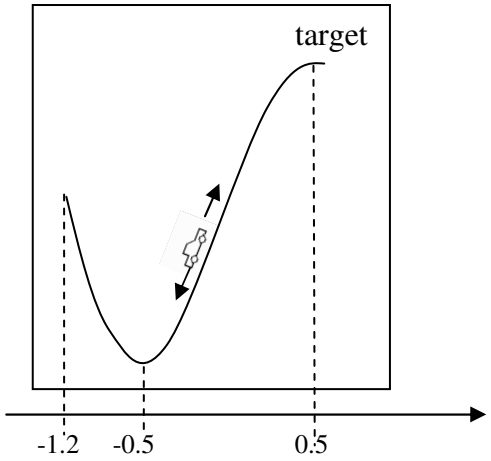


Figure 7 - mountain car problem environment.

The problem may be described as follows: a car must be able to get to the top of a mountain; however, the car is not powerful enough to overcome the force of gravity. Thus, in order to achieve its objective, the car must begin by moving in the opposite direction of the target so that it may add the acceleration of gravity to its own acceleration (see Figure 7).
The problem has two continuous state variables, the position of the car $x_t \in$ [-1.2, 0.5] and its velocity $v_t \in$ [-0.07, 0.07], and three discrete actions: an impulse to the left (F_t = -1); no impulse (F_t = 0); and an impulse to the right (F_t = 1). The system's dynamics is described by equation 10.

The learning process starts at -0.5 and -1.2 position with zero velocity alternating at each cycle of the experiment and tests were carried out with variations in the initial conditions (position and velocity). The purpose of these tests was to evaluate the performance of the models in several different situations.

Table 1 compares the results from the learning and testing phases of the RL-HNFP models with different RL-based models described in [Jouffe, 1998].

Table 1. Performance comparison of the RL-HNFP and extended RL-HNFP models with other models [Jouffe, 1998] for the mountain-car problem.

Model	No. of parameters	Learning Phase	Testing Phase
Neural-Q-learning	4	1724	2189
CMAC Q-learning	343	262	85
FQL	25	112	61
RL-HNFP	0	91	69
RL-HNFP$^+$	0	88	72

The first column in Table 1 indicates the model being evaluated and the second indicates the number of parameters, that is, the number of neurons in the hidden layer for the Neural Q-Learning case, the number of grids for the CMAC (Cerebellar Model Articulation Controller) case, and the number of rules for the FQL (Fuzzy Q_Learning). The third and fourth columns display the performance obtained for each model in terms of number of steps. The fourth columns indicate the average performance obtained for each model in terms of number of steps with the car starting out from any x $\in$ [-1.2,0.5] and v $\in$ [-0.07,0.07].

The Neural Q-Learning model presents the worst result because, in addition to learning the Q-values, it also needs to learn to identify the states; the modifications that the backpropagation algorithm makes in the weights affect the entire network and are detrimental to the learning process.

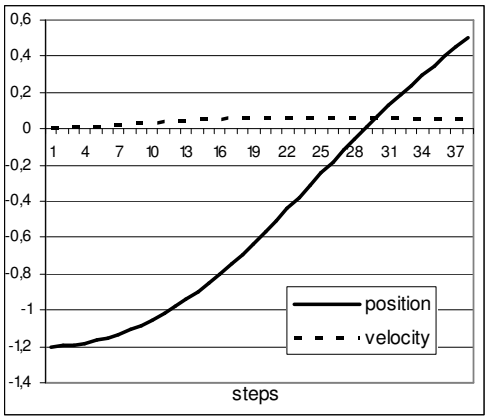
In the case of the CMAC model, the active state set is perceived discretely (many neighboring states activate the same grid sets); on the other hand, in the case of the FQL and RL-HNFP models, the state set is perceived continuously. This explains why the results of the FQL and RL-HNFP models are better [Jouffe, 1998].

Despite the FQL results being slightly better than the result obtained with the RL-HNFP and RL-HNFP$^+$ models, the former (as well as the CMAC) starts from a priori information (fuzzy rules and sets) relative to the learning process. Additionally,

$$v_{t+1} = \min(0.07, \max(-0.07, v_t + 0.001F_t - 0.0025\cos(3x_t)))$$
$$x_{t+1} = \min(0.5, \max(-1.2, x_t + v_{t+1})) \quad (10)$$

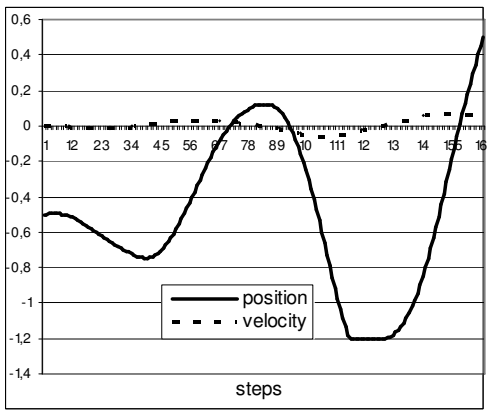
although RL-HNFP and RL-HNFP$^+$ models use a more elaborated return function than FQL model, this function is related with the agent's objective, not with the learning process.

Figures 8 and 9 depict the results obtained with the RL-HNFP model and Figures 10 and 11 with the extended model RL-HNFP$^+$.



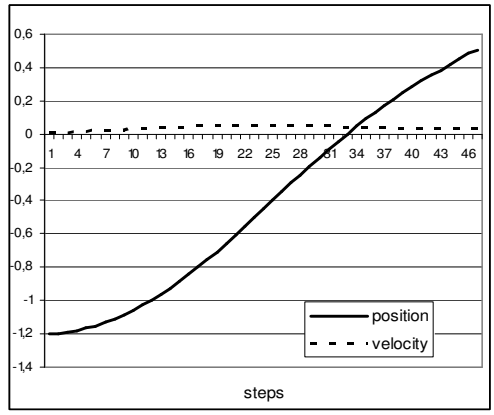
initial position =-1.2 initial velocity = 0
Steps = 38

Figure 8 - Test result for the *mountain car problem starting from x = -1.2* with the RL-HNFP model



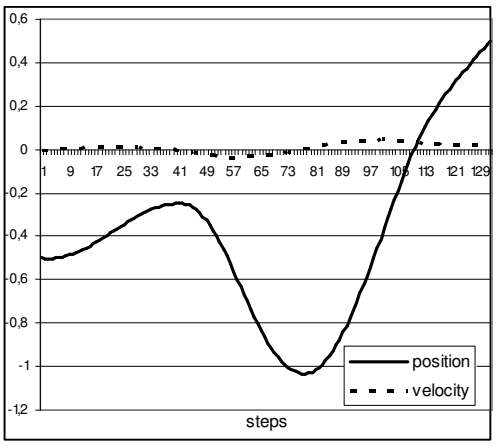
initial position =- 0.5 initial velocity = 0
Steps = 166

Figure 9 - Test result for the *mountain car problem starting from x = - 0.5* with the RL-HNFP model



initial position =-1.2 initial velocity = 0
Steps = 46

Figure 10 - Test result for the *mountain car problem* with the extended RL-HNFP model



initial position =-0.5 initial velocity = 0
Steps = 130

Figure 11 - Test result for the *mountain car problem* with the extended RL-HNFP model

The most favorable initial condition for the car is when it is in the position -1.2 (Figures 8 and 10). Figures 8 and 10 show that the car needs more steps to reach its objective using RL-HNFP$^+$ model than with the original model. One of the reasons is related to the resultant RL-HNFP$^+$ smaller structure (see table 2).

The most unfavorable initial condition for the car is when it is in the position -0.5 with velocity 0 (Figures 9 and 11). Figures 9 and 11 show that the car needs more steps to reach its objective using RL-HNFP$^+$ model than with the original model.

However, illustrations 9 and 11 show that the model RL-HNFP$^+$ didn't need to go to the end opposite of the environment (position -1.2) to reach the objective, concluding the task with smaller number of steps.

It may be observed from the graphs that when the car starts out from the valley (position -0.5), it needs to oscillate from one side to the other so as to gain enough momentum to overcome the force of gravity and reach the "mountaintop". On the other hand, when the car starts out from the farthest position (position -1.2), it already has enough potential energy to reach the objective without needing to oscillate.

Table 2 presents the average number of cycles and number of cells for the RL-HNFP extended RL-HNFP.

Tables 1 and 2 demonstrates that the RL-HNFP$^+$ model needs less cycles to accomplish the learning process and uses a smaller structure than the structure of the original model, reducing the computational time (due to the size of the structure and to the number of cycles).

Table 2. Performance comparison of the RL-HNFP model and extended RL-HNP (RL-HNFP$^+$) for the mountain-car problem.

Model	Cycles	Cells
RL-HNFP	5000	96
RL-HNFP$^+$	3000	15

5. CONCLUSIONS

This paper presented a new class of neuro-fuzzy models, called Reinforcement Learning Hierarchical Neuro-Fuzzy Systems, which improve the weak points of conventional neuro-fuzzy systems.

The RL-HNFP models are able to: create and expand the structure of rules without any prior knowledge (fuzzy rules or sets); extract knowledge from the agent's direct interaction with large and/or continuous environments (through reinforcement learning); and produce interpretable fuzzy rules, which compose the agent's intelligence to achieve its goal(s).

As demonstrated bye the case study, the agent was able to generalize its actions, showing adequate behavior when it was in states whose actions had not been specifically learned. This capacity increases the agent's autonomy.

This experiment demonstrates that these models are suitable for control problems, presenting good generalization and generating their own hierarchical structure of rules with linguistic interpretation. Moreover, the automatic environment learning endows the agent with intelligence (knowledge base, reasoning and learning). These are characteristics that increase the autonomous capacity of this agent.

The proposed modifications, relative to the learning algorithm, allowed a faster learning (with a smaller number of learning cycles), a more compact structure and a lower computational cost.

In spite of the good results obtained by the proposed model, a few points still need improvement. The authors are developing a new model, called Multiagents Reinforcement Learning – Neuro-Fuzzy Politree (MARL-HNFP), for other types of applications that demand the use of multiple agents.

REFERENCES

Abraham A. 2005, "Adaptation of Fuzzy Inference System Using Neural Learning", Fuzzy System Engineering: Theory and Practice, Nadia Nedjah et al. (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, ISBN 3-540-25322-X, Chapter 3, pp. 53-83.

Bishop C.M. 1995, "Neural Networks for Pattern Recognition", Clarendon Press -Oxford, 1995.

Boyan J.A. and Moore A.W. 1995, "Generalization in reinforcement learning: Safely approximating the value function", in: G. Tesauro, D. S. Touretzky, and T. K. Leen, (Eds.), Advances in Neural Information Processing Systems 7, Cambridge, MA, The MIT Press.

Figueiredo K., Vellasco M.M.B.R., Pacheco M.A.C. and Souza F.J. 2004, "Reinforcement Learning-Hierarchical Neuro-Fuzzy Politree Model for Control of Autonomous Agents", Fourth International Conference on Hybrid Intelligent Systems (HIS'04), (ISBN 0-7695-2291-2), IEEE Computer Society, Kitakyushu, Japan, December, pp.130-135.

Figueiredo K., Vellasco M.M.B.R. and Pacheco M.A. 2005, "Hierarchical Neuro-Fuzzy Models based on Reinforcement Learning for Intelligent Agents", Lecture Notes in Computer Science - Computational Intelligence and Bioinspired Systems, Volume 3512, Editors: Joan Cabestany, Alberto Prieto, Francisco Sandoval, (ISBN: 3-540-26208-3), Proc. of The 8th International Work-Conference on Artificial Neural Networks (IWANN'2005), Barcelona, Spain, June 8-10, pp. 424-431.

Finkel R. and Bentley J. 1974, "Quad trees, a data structure for retrieval on composite keys", Acta Informatica 4, pp. 1-9.

Gonçalves L.B., Vellasco M.M.B.R., Pacheco M.A.C. and Souza F.J. 2004, "Inverted hierarchical neuro-fuzzy BSP system: A novel neuro-fuzzy model for pattern classification and rule extraction in databases", accepted for publication in, IEEE Trans. on Systems, Man and Cybernetics, Part C).

Haykin S. 1998, "Neural Networks – A Comprehensive Foundation", Macmillan College Publishing Company, Inc.

[Jang, 1993] Jang J.S.R. 1993. "ANFIS: Adaptive-network-based fuzzy inference system", IEEE Transactions on SMC, 23, No. 3, pp. 665-685.

Jang J.-S. R., Sun C.-T., E. 1997, "Mizutani, Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence", Prentice-Hall.

Jouffe L. 1998, "Fuzzy inference system learning by reinforcement methods", IEEE Trans. on Systems, Man and Cybernetics, part c, 28, No. 3, pp. 338-355.

Klir G.J. and Yuan B. 1995, "Fuzzy Sets and Fuzzy

Logic – Theory and Applications", Prentice Hall PTR.

Kruse R. and Nauck D. 1995, "NEFCLASS-A neuro-fuzzy approach for the classification of data", Proc. of the 1995 ACM Symposium on Applied Computing, Nashville, 26-28.

Lin C.T. and Lee C.S.G. 1996, "Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems", Prentice-Hall.

Mendel J.M. 1995, "Fuzzy logic systems for engineering: A tutorial", Proceedings of the IEEE, 83, No. 3, pp. 345-377.

Moore A.W. 1991, "Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces", Proceedings of the Eighth International Conference on Machine Learning, in: Birnbaum and G.C.Collins, (Eds.), Morgan Kaufmann, pp. 333-337.

Souza F.J., Vellasco M.M.B.R. and Pacheco M.A.C., 2002a, "Hierarchical neuro-fuzzy quadtree models", Fuzzy Sets and Sys, 130/2, pp. 189-205.

Souza F.J., Vellasco M.M.B.R., Pacheco M.A.C. 2002b, "Load Forecasting with The Hierarchical Neuro-Fuzzy Binary Space Partitioning Model", International Journal of Computers Systems and Signals, (ISSN 1608-5655), International Association for the Advancement of Methods for System Analysis and Design (IAAMSAD), South Africa Vol. 3, No. 2, pp. 118-132.

Sutton R. S. 1996, "Generalization in Reinforcement learning: Successful examples using sparse coarse coding", Advances in Neural Information Process Systems 8, pp. 1038-1044, MIT Press.

Sutton R.S. and Barto A.G. 1998, "Reinforcement Learning: An Introduction", MIT Press.

Vellasco M.M.B.R., Pacheco M.A.C., Ribeiro Neto L.S., Souza F.J. 2004, "Electric Load Forecasting: Evaluating the Novel Hierarchical Neuro-Fuzzy BSP Model", International Journal of Electrical Power & Energy Systems, (ISSN 0142-0615), Vol. 26, No. 2, pp. 131-142, Elsevier Science Ltd, February.

Vuorimaa P. 1994, "Fuzzy self-organizing map", Fuzzy Sets and Systems 66, pp. 223-231.

BIOGRAPHY

Karla Figueiredo received the B.Sc. degrees in electrical engineering from Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil, in 1989., and M.Sc. and D.Sc. degrees in electrical engineering from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil, in 1994 and 2003, respectively. In 2004, she became a visiting professor at the Electronics and Telecommunications Engineering Department of Rio de Janeiro State University. Her main research interests include Neuro-Fuzzy Systems, Reinforcement Learning and Multiagents Systems.

Marcio Luiz Ramos dos Santos received the B.Sc. degree in electronical engineering in 1994 from Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, and the M.Sc. degree in production engineering in 2000 from Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro. In 2002, he started doctoral studies in electrical engineering at Pontifical Catholic University of Rio de Janeiro. He is currently a project manager at DBA System Engineer (supplier of IT solutions). His research interests include Neuro-Fuzzy Systems, Reinforcement Learning and Multiagents Systems.

Marley Maria Bernardes Rebuzzi Vellasco (M'89) received the B.Sc. and M.Sc. degrees in electrical engineering from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil, in 1984 and 1987, respectively, and the Ph.D. degree in computer science from the University College London (UCL), London, U.K., in 1992. She is currently an Assistant Professor at the Electrical Engineering Department of PUC-Rio and jointly heads the Applied Computational Intelligence Laboratory (ICA) of PUC-Rio with Dr. M. Pacheco. She is also an Assistant Professor at the Computing and Systems Engineering Department, State University of Rio de Janiero (UERJ). She is the author of over 100 papers in books, professional journals, and conference proceedings in the area of soft computing. Her research interests include neural networks, fuzzy logic, neuro-fuzzy systems and evolutionary computation for decision support systems, pattern classification, time-series forecasting, control, optimization, knowledge discovery databases, and data mining.

Marco Aurélio Cavalcanti Pacheco received the B.Sc. and M.Sc. degrees in electrical engineering from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil, in 1976 and 1980, respectively, and the Ph.D. degree in computer science from the University College London (UCL), London, U.K., in 1991. He is currently an Assistant Professor at the Electrical Engineering Department of PUC-Rio and jointly heads the Applied Computational Intelligence Laboratory (ICA) of PUC-Rio with Dr. M. Vellasco. He is the author of over 100 papers in books, professional journals, and conference proceedings in the area of soft computing. His research interests include evolutionary computation, evolvable hardware, nanotechnology, neural networks, fuzzy systems, applied computational intelligence, knowledge discovery databases, and data mining.