Reinforcement Learning Hierarchical Neuro-Fuzzy Model for Autonomous Robots

Priscila Dias[‡], Karla Figueiredo[£],
Marley Vellasco[‡†§], Marco Aurélio Pacheco[‡†], Carlos H. Barbosa[‡]

[‡]Department of Electrical Engineering
Pontifícia Universidade Católica do Rio de Janeiro, Brasil

[£]Department of Electronics/Telecommunications Engineering
Universidade Estadual do Rio de Janeiro, Brasil

[†]Department of Systems and Computing Engineering
Universidade Estadual do Rio de Janeiro, Brasil

[§]Department of Computer Science
University College London, UK

cpdias@gbl.com.br, karlaf@uerj.br, [marley, marco, hall]@ele.puc-rio.br

Abstract

This work presents the application of a new hybrid neuro-fuzzy model, called RL-HNFP (Reinforcement Learning – Neuro Fuzzy Hierarchical Politree), in automatic learning of autonomous robots. This model provides an agent with intelligence, making it capable, by interacting with its environment, to acquire and retain knowledge for reasoning (infer an action). Promising results have already been obtained in tests with a Khepera robot simulator in a simplified environment (low complexity). The objective of this work is to evaluate the RL-HNFP model performance in a bigger and/or more complex environment, carrying out the necessary modifications in the original model to achieve good performance in this new environment. The adapted system was evaluated again with Khepera and the obtained results show the potential of this modified model to efficiently interact in more elaborated situations.

Keywords: Intelligent Agent; Neuro-Fuzzy Systems; Reinforcement Learning; Automation; Hierarchical Partitioning; Robotics.

1 Introduction

Nowadays, many applications involve unknown environments that should be explored and recognized in order to achieve a fixed goal. Attempting to deal with these kinds of applications, this work has the aim to evaluate the new *Reinforcement Learning - Neuro-Fuzzy Hierarchical Politree* model (RL-NFHP) [1] in a bigger and/or more complex environment, making the appropriate adaptations to its correct execution.

The original RL-NFHP model [1] was evaluated in four applications: mountain car, inverted pendulum, cart centering problem, and Khepera robot control [2]. In the Khepera case study, the objective was to control a robot that leaves an initial position in a fixed environment and reaches another position defined as its objective, considering an obstacle in the center of the environment.

In the extension proposed in this paper, the intention is to lead the robot to the same goal, but in a bigger and/or more complex environment. Therefore, some adaptations were implemented in the original model, as well as modifications in the robot's simulator (developed inside the reinforcement learning algorithm).

This paper has been organized in 4 additional sections. Section 2 describes the original RL-HNFP model as well as the proposed changes to allow it to be used in bigger and/or more complex environments. The case studies are presented in section 3. Finally, sections 4 and 5 present, respectively, the results obtained and the conclusions.

2 RL-HNFP Model

The RL-HNFP is a hybrid neuro-fuzzy hierarchical model, based on reinforcement learning, that was conceived from the analysis of the limitations in the existing neuro-fuzzy models (such as: the

[§] Marley Vellasco is currently at the Dept. of Computer Science, UCL, on a Postdoctoral Program sponsored by CAPES - Brazil.

predefinition of the number and format of the membership functions used in the fuzzy rule's antecedents and consequents; the predefinition of the number of rules as well as the specification of their antecedents; and the limitation of the number of inputs in the model) [3-6] and of the desirable characteristics for reinforcement learning based systems, particularly in applications involving continuous environments and/or environments of high dimension. These environments present a characteristic known as curse of dimensionality [3] that makes unfeasible the use of traditional methods of reinforcement learning.

The use of recursive partitioning methods (already explored with excellent results in [7] and [8]), which significantly reduces the limitations of the existing neuro-fuzzy systems, was of fundamental importance to reach the model's objectives. The use of a recursive partitioning method combined with reinforcement learning resulted in a new class of Neuro-Fuzzy Systems (NFS), called *Reinforcement Learning-Neuro-Fuzzy Hierarchical System* (RL-NFH). Therefore, the new RL-NFH 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 the action that should 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 the traditional neuro-fuzzy systems; and it automatically generates linguistic rules.

In this new neuro-fuzzy model, the state identification process, which is not previously known, is accomplished by the learning phase. The generation of rules is obtained with an automatic partitioning process of the input space, as will be described in section 2.3.

The reinforcement learning process is based on the SARSA method [3]. It is an algorithm based on Temporal Difference that does not need modeling of the environment and can be incremental. The function value Q is defined as being the sum of the expected values of the reinforcements obtained by the execution of action a in state s, in accordance with a policy π; its value can be updated according to equation (1).

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma.Q(s',a') - Q(s,a)] \qquad (1)$$

The $Q(s,a)$ value is updated based on its current value $Q(s,a)$; r is the immediate local reinforcement; γ is a parameter that specifies a percentage of the contribution of the Q-value associated with the next action a', chosen when system is in state s' ($Q(s',a')$), and α is the parameter that is proportional to the relative contribution of this local action within the global action.

The functionality of the model's fuzzy component is to aggregate states that have similar behaviors and associate them with the same action. The reinforcement-learning component makes the model 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, in turn, may have as consequent a subsystem with the same structure. This characteristic smoothens the value function generalization process without affecting the results, which is a good requirement for process convergence [9].

The RL-HNFP model is made up of one or several standard cells 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 subsections briefly describe the main characteristics of this new neuro-fuzzy model: the partitioning process, the basis cell and the learning algorithm.

2.1 Hierarchical Partitioning

The partitioning process of the input/output space has great influence on the performance of a neuro-fuzzy system in relation to its desirable features (accuracy, generalization, automatic generation of rules etc.).

The politree partitioning was inspired by the quadtree structures proposed in [7] and [10]. The latter one works only with two-dimensional spaces, while the first works with n-dimensional spaces. In the politree partitioning, the space is recursively and successively subdivided in 2^n subspaces (for **n** inputs). The politree partitioning can be represented by a tree structure as shown in figure 1.

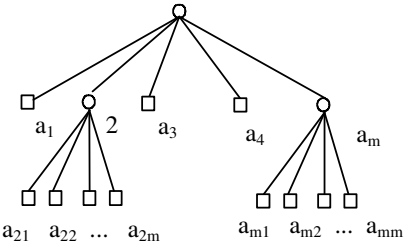


Figure 1: Politree partitioning.

The Politree 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. This type of partitioning is flexible and minimizes the problem of exponential growth of rules, since it creates new rules locally according to the learning process described in section 2.3.

2.2 Basic Cell

An RL-NFP cell is a mini-neuro-fuzzy system that performs politree partitioning in a given space, using the membership functions ρ and μ, referred to the *low* and *high* fuzzy sets of each input variable (implemented through complementary sigmoids). Each cell has all the inputs that are being considered in the problem.

Figure 2 below presents an illustration of the cell's representation in a two-dimensional input space (Quadtree), producing a simpler drawing than the n-dimensional form proposed by Politree.

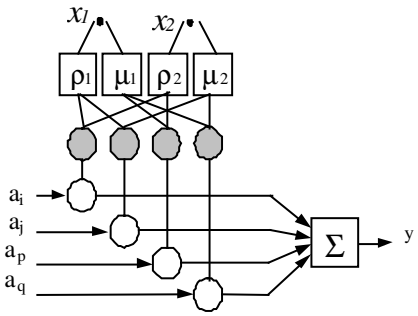


Figure 2: Basic RL-NFP cell with 2 inputs.

As can be observed, rules antecedents are defined by input variables associated to fuzzy sets. The values of these variables are read by the agent's sensors and then are inferred in the antecedent's fuzzy sets (ρ_i e μ_i). If the antecedent is true, the rule is fired.

The consequents are actions (a_i) that the agent must learn during the process and that will be performed by its actuator. Therefore, the RL-HNFP model creates and determines its structure by mapping states into actions.

The linguistic interpretation of the mapping implemented by the RL-NFP cell of figure 2 is given by the following set of rules:

If $x_1 \in \rho_1$ and $x_2 \in \rho_2$ then $y = a_i$.

If $x_1 \in \rho_1$ and $x_2 \in \mu_2$ then $y = a_j$.

If $x_1 \in \mu_1$ and $x_2 \in \rho_2$ then $y = a_p$.

If $x_1 \in \mu_1$ and $x_2 \in \mu_2$ then $y = a_q$.

The consequents of the cell's 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.

According to figure 3, each action a_i is associated with a function value *Q*. By the reinforcement learning method, one action of each poli-partition (for example, a_i, a_j, a_p and a_q in figure 2) will be defined as the one that represents the desired behavior of the system whenever the system is in a given state. A state defines the active cells and each poli-partition of each cell selects its own action.

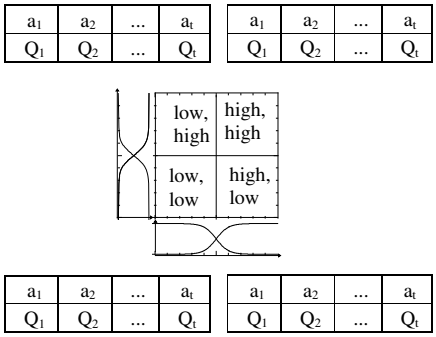


Figure 3: Internal representation of the RL-NFP cell with two inputs.

When singletons are used as fuzzy rules consequents, the most suitable defuzzification method is the weighted average, since it combines the consequents of the fuzzy rules (actions a_i) with each of these rules firing level (α_i), thereby generating an output (crisp) 'y' according to the expression given in equation (2). The denominator in this equation is equal to one for any value of x_1 and x_2, due to the usage of complementary membership functions (high (μ) and low (ρ)).

$$y = \left(\frac{\sum_{i=1}^{4} \alpha_i \times a_i}{\sum_{i=1}^{4} \alpha_i} \right) = \sum_{i=1}^{4} \alpha_i \times a_i \qquad (2)$$

Where:

$$\alpha_1 = \rho_1 (x_1) * \rho_2 (x_2)$$

$$\alpha_2 = \rho_1 (x_1) * \mu_2 (x_2)$$

$$\alpha_3 = \mu_1 (x_1) * \rho_2 (x_2)$$

$$\alpha_4 = \mu_1 (x_1) * \mu_2 (x_2)$$

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.

2.3 Learning Algorithm

The RL-HNFP model performs two learning tasks in a single algorithm (structure identification and parameter adjustment).

The learning process starts out with the definition of the input variables that are relevant to the environment in which the agent is inserted. Two other definitions must also be provided: the set of actions

the agent may use in order to attain its objectives; and the initial *Q*-values associated with these actions (figure 3).

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

The flowchart shown in figure 4 presents the learning algorithm of the model with its 6 most relevant steps. In the sequence these steps are briefly described.

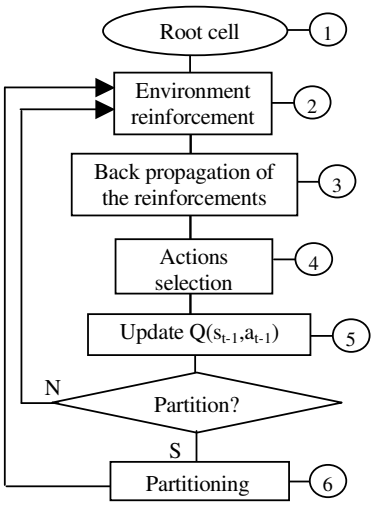


Figure 4: RL-HNFP algorithm.

Step 1 – Root cell: A root-cell is created with fuzzy sets whose domain is the universe of discourse of the cells input variables. These variables are read from the environment, normalized, applied to the cells input, and evaluated in the cells high and low fuzzy sets. Each poli-partition chooses one of the possible actions, based on the methods described in step 4 of this algorithm. The defuzzification process presented in section 2.2 calculates the cell's output.

Step 2 – Environment reinforcement: After the resultant action is carried out, the environment is read once again. This reading enables calculation of the environment reinforcement value, which will be used to evaluate the action taken by the agent by means of an evaluation function.

Step 3 – Back propagation of the reinforcements: At each step, the reinforcement is calculated for each partition of all active cells, through its participation in the resulting action. This occurs via the back propagation of the global reinforcement from the root-cell until the leaf-cells poli-partitions. Thus, a local reinforcement value is defined for each poli-partition.

Step 4 – Actions selection: The actions experimented during the reinforcement learning are selected from a set of actions. Each of them is associated to a *Q*-value function. The model uses a greedy policy [3], which alternates actions associated to the highest expected *Q*-value with randomly selected actions.

Step 5 – Update Q(s_{t-1},a_{t-1}): This update is based on the evaluation between the current and previous global reinforcements and takes place in two distinct forms. When the current global reinforcement value is higher than the previous global reinforcement, the system will be rewarded. Otherwise, it will be punished.

Step 6 – Partitioning: The partitioning is implemented through a growth variable and a growth function (defined in terms of the algorithm number of cycles and steps) created to allow or prevent structure growth. According to the relation between the current local reinforcement and the previous local reinforcement, as well as the percentage of variation in the *Q*-value functions of actions associated with any of the poli-partitions of an active cell, the growth variable is incremented or reinitialized. When this variable value is greater than the evaluated growth function value, partitioning is executed, that is, a leaf cell is created and connected to that poli-partition.

2.4 RL-HNFP Model Extension

The aim of this work was to evaluate the RL-HNFP model performance in a bigger and/or more complex environment. Consequently, some adaptations were necessary so that the system could still provide good performance. These modifications led to a more compact and efficient structure, producing results even better than previously expected.

The growth variable increase/decrease values were modified in respect to *Q*-value function variation. When the current local reinforcement is smaller than the previous local reinforcement, the growth variable should be decreased instead of reinitialized. The obtained results were very relevant to prevent excessive and unnecessary structure growth.

Limitation of cells domain is another way to avoid structure growth. Since this domain (except for the root cell) is the sub-domain corresponding to the low or high bi-partition of the cell's nearest ancestral, the partitioning limitation can be implemented by establishing a minimum fraction of the domain that can be reached, beyond which the cell cannot be split. In this new proposed model this fraction has been increased.

The alpha-cut parameter in the original model was also increased. This parameter is used to discard rules whose activation degree is lower than its value. This parameter is important to restrict the number of active cells, which allows the learning process to concentrate on more relevant cells.

Usually, an agent chooses actions that are related to good previous reinforcements; that is, it tries to

exploit what has already been learned (*exploitation*). However, to find better actions, it is necessary to search for alternatives that might be better than the ones already experimented; that is, the agent needs to explore the environment (*exploration*). So these two policies can be joined together to achieve an optimal solution. This is done by means of a probabilistic method where each action has a chance of being randomly selected; as a result exploration and exploitation factors are merged.

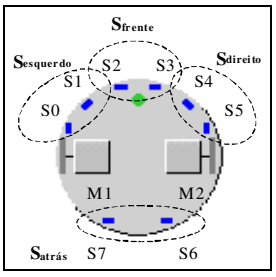
At first, the initial probability of choosing actions randomly should not be very restricted/small, because in this manner the agent can have difficulties to find the better path to reach its objective. Hence, this probability was increased and, then, modified according to the agent's evolution.

In addition, the consequent of the poli-partition with highest contribution to a bad action will be, necessarily, randomly chosen in the next action. In this way, the system tries to avoid the continuous selection of this bad action.

3 Case Study

The RL-HNFP model, together with a Khepera robot simulator, was originally [1] tested in a squared environment where the agent moved from one of the corners to reach the diametric opposite corner. Nevertheless, he could not pass through the center of the environment because of an obstacle. In this work the objective is to extend the original model, so that it can be tested in bigger and/or more complex environments.

The robot acquires signs from the environment using 8 sensors grouped into 4: one ahead, one in each side and one behind. Its actions are executed via 2 independent motors with power varying between -20 and 20, one in the right side and the other in the left side. The robot scheme can be seen in figure 5; the relation among angles can be seen in figure 6.



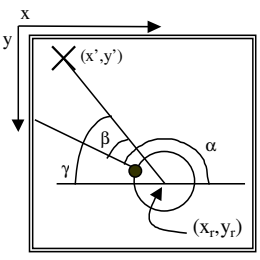


Figure 5: Khepera robot. **Figure 6:** Angles definition.

The simulator is composed of the robot and the navigation environment, defined by limits (walls) and by obstacles that can exist or not. It provides the robot's position (x,y) and the angle α between the front of the robot and x axis (figure 6). These data are calculated in accordance with robot's rotation and translation movements and chosen action.

Angles β and γ, between the front of the robot and the goal and between x axis and the goal, respectively, are calculated from the information given by the simulator and are used throughout the process.

The robot frontal sensor has an adjustable field of view, so that it can have a wider sight of the environment. The obstacles treatment is done in a generic way, based on their characteristic points.

Sensors sensibility is defined by means of exponential functions that consider the normalized Euclidian distance between sensor and obstacle.

Since this work is based on a simulator, not a real robot in a real environment, some other important issues had to be addressed, such as avoiding the robot passing through the obstacles or going beyond the environment limits.

4 Experimental Results

The proposed changes were tested in two distinct situations: firstly, in an environment without obstacles; then in an environment with a central obstacle like in the original model. In both cases, however, the environment complexity was greater than before, due to the increase in its size (3 times larger than originally). The results obtained are described in the following sections.

4.1 Environment without Obstacle

One of the tests carried out without obstacles is shown in figure 7, where the black dot represents the front of the robot. With a structure generated only from position (-5, -5) with angles 0º, 90º, 135º and -135º, tests were performed from positions (-5, -5, 45º), (-5, -5, -135º), (-5, 5, -90º) and (5, -5, 0º). The robot's objective was to reach position (5, 5). It can be observed that the second and forth testing positions lead the robot directly to the walls, demonstrating that the acquired knowledge was generalized and that the obtained results conform to the expectation.

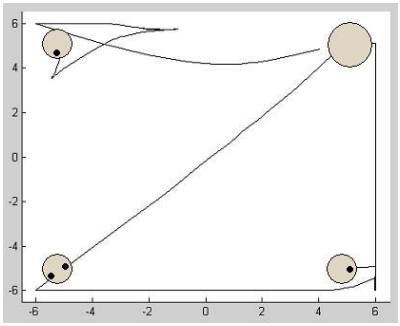


Figure 7: Tests without obstacles.

4.2 Environment with Central Obstacle

The results described in figure 8 refer to tests executed with a structure trained from a variety of positions different from the ones in the tests. Once

more, the results indicate the good performance of the model.

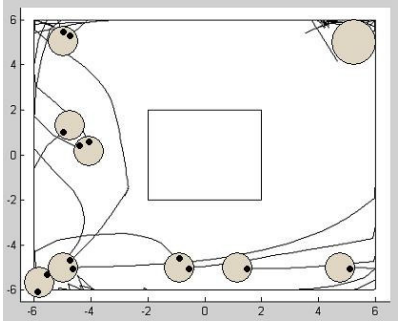


Figure 8: Central obstacle tests.

Table 1 presents the average number of cycles and the number of cells of the original model as well as of the modified model. It is important to stress that the proposed model was evaluated in an environment that is 3 times greater than the one in the original model, resulting in a more complex environment.

Table 1: Original model x proposed model.

	Without Obstacle		**Central Obstacle**	
	Cycles	*Cells*	*Cycles*	*Cells*
Original Model	5.000	498	10.000	812
Proposed Model	1.000	19	5.000	40

Even though the RL-HNFP model has not been submitted to a real robot, it seems that the necessary computational effort would be small and the process very fast, once a set of hierarchical rules could be extracted from the generated structure at the end of the learning procedure. In this way, at each step, the robot could refer to these rules and execute the resulting action.

5 Conclusions

This article presented an extension of a neuro-fuzzy hierarchical model, called RL-HNFP, capable of: creating and expanding the structure of rules without any prior knowledge (fuzzy rules or fuzzy sets); extracting knowledge from agent's direct interaction with large and/or continuous environments through reinforcement learning; and producing interpretable knowledge, by means of fuzzy rules, which constitute the agent's intelligence to achieve its goal(s). The model was able to generalize its actions, showing adequate behaviour when the agent was in states whose actions had not been specifically learned. This capacity increases the agent's autonomy.

The proposed modifications, relative to the learning algorithm, allowed a faster learning, a more compact structure and a lower computational cost.

Tests have shown that even in bigger environments, the RL-HNFP model converges and reaches efficiently the expected result. The case studies confirm the successful application of the model in automation and robotics, what encourages further research on automatic learning using Hierarchical Neuro-Fuzzy Systems.

In future the authors intend to execute tests with *Eligibility Traces* [3]. This is a method that does not update only the current state function value, but also the function value of previous states inside a predefined limit. Although some tests were implemented with this resource, conclusive results were not obtained; so more tests to better evaluate its effects are still necessary.

6 References

[1] K. Figueiredo. (2003) "Agentes Inteligentes: Novos Modelos Neuro-Fuzzy Hierárquicos com Aprendizado Automático baseado em Reinforcement Learning". *PhD Thesis* – DEE, PUC-Rio, Brazil – in Portuguese.

[2] E. Ostergard. (2000). "Evolving Complex Robot Behaviour", *MSc Thesis, Dept. Computer Science*, University of Aarhus, Denmark.

[3] R. S. Sutton & A. Barto. (1998) *Reinforcement Learning: An Introduction*, MIT Press.

[4] W. Moore. (1991). Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces, *Proc of the 8th Int. Conf. on Machine Learning*, Morgan Kaufmann, 333-337.

[5] J. Boyan and A. Moore. (1995). Generalization in reinforcement learning: Safely approximating the value function, G. Tesauro, D. S. Touretzky, and T Leen, eds, *Advances in Neural Information Processing Systems* 7, MIT Press.

[6] L. Jouffe. (1998). Fuzzy Inference System Learning by Reinforcement Methods, *IEEE Trans on Systems, Man & Cybernetics-C*, 28/3,p.338-355.

[7] F. Souza, M. Vellasco and M. Pacheco. (2002). Hierarchical Neuro-Fuzzy QuadTree Models, Fuzzy Sets & Systems, vol. 130/2, pp. 189-205, August 2002, Elsevier Science.

[8] M. Vellasco, M. Pacheco, L. Ribeiro Neto and F. Souza. (2003b). Electric Load Forecasting: Evaluating the Novel Hierarchical Neuro-Fuzzy BSP Model, *International Journal of Electrical Power & Energy Systems*, (ISSN 0731-356X), Vol. 26, pp. 131-142, Elsevier Science Ltd.

[9] Sutton, R.S., Generalization in Reinforcement learning: Successful examples using sparse coarse coding, In Touretzky, D.S., Mozer, M.C., and Hasselmo, M.E., editors, Advances in Neural Information Processing Systems 8, pp. 1038-1044, MIT Press, 1996.

[10] R. A. Finkel and J. L. Bentley. (1974). Quad trees, a data structure for retrieval on composite keys. *Acta Informatics 4*, pp. 1-9.