10th Brazilian Congress on Computational Intelligence (CBIC'2011), November 8 to 11, 2011, Fortaleza, Ceará Brazil

BUILDING MULTI-AGENT SYSTEMS WITH REINFORCEMENT HIERARCHICAL NEURO-FUZZY MODELS

Marcelo França Corrêa[1], Marley Vellasco[1] and Karla Figueiredo[1,2]

[1] Depto. Eng. Elétrica, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
[2] Núcleo de Computação Científica, CCMAT, Universidade Estadual da Zona Oeste - UEZO
marcelo_correa@hotmail.com; marley@ele.puc-rio.br ; karlafigueiredo@uezo.rj.gov.br

Abstract – Multi-Agent Systems (MAS) are an emergent area of Computational Intelligence that provides the tools for the construction of complex systems involving multiple agents and the coordination mechanisms amongst them. Because of its properties, MAS has been used into a great variety of fields, such as distributed control, robotic teams, navigation control, route planning control, lift programming, cargo balancing, automatic trading among agents, etc. This paper presents a new MAS that works though hierarchical neuro-fuzzy hybrid models. The main advantage of this class of system is the autonomous capacity to create rules, expand their own rule structure, and extract knowledge from the direct interaction between the agent and the environment. The new model was tested through the Pursuit game benchmark application. Some preliminary results have shown promising. The tests demonstrated that the developed system has shown a good capacity of convergence and coordination among the intelligent agents.

Keywords – Multi-Agent Systems (MAS), hierarchical neuro-fuzzy, intelligent agents, reinforcement learning

1 Introduction

Conventionally, actions executed by computers must be predicted, planned and encoded by programmers. In the case of unexpected situations, it is configured an error that must be repaired by the programmer. This is quite acceptable for most systems. However, for a growing number of applications, systems must have a higher autonomy level. They should be aware of what must be done to satisfy the goals that were determined by the programmer (Weiss, 1999). Intelligent agents are the paradigm used in the development of software applications that aim at filling this need. They perceive the environment using sensors and act through actuators (Jenning & Wooldridge, 1997).

Amongst the many characteristics that an agent can hold, autonomy is the one that determines that programs (codes or algorithms) can be classified as agents. Autonomy is the agent's capacity to follow its goals automatically, meaning without interactions or commands from the environment. Another really important characteristic, as it raises the autonomous capacity of an agent, is the intelligence. Brenner et. Al. (1998) are emphatic when stating that the agent must be capable of learning, so it can be considered autonomous. Learning, along with the knowledge basis and the rationality, compose the characteristic "intelligence".

An intelligent agent must interact with the environment in order to reach its goals; it must be capable of gathering information from its environment, deciding based on the aforementioned information and starting a specific action based on these decisions.

Recently, a new research field called Multi-Agent Systems (MAS) has been emerging, aiming at developing complex systems involving multiple agents and the coordination mechanisms among them. A MAS can be defined as a group of autonomous agents, interacting amongst themselves and sharing the same environment, which is perceived through sensors, and where they act performing actions (Vlassis, 2003). This type of system has been finding its way into a great variety of fields, such as distributed control, robotic teams, navigation control, route planning control, etc.

The benefits brought by the application of multiple agents are many. First, through parallel computing, several agents can work together to better exploit the decentralized structure of a certain task and speed up its conclusion (Kok et al., 2005; Fitch et al., 2005). Besides that, agents can share experiences through communication (Tan, 1993), they can just observe the skillful ones and learn (Price e Boutilier, 2003), the most intelligent ones can be teachers (Clouse, 1995), etc. The MAS can also provide a high degree of scalability, through the addition of new agents when necessary, and still make agents take over others in cases of failure (Busoniu, 2008). And it can help in the study and elucidation of the "intelligence" (Weiss, 1999). As "intelligence" is tightly attached to interaction, the best way for developing intelligent machines may be through the development of "machines' social networks".

Several models of agents that have been developed so far use Reinforcement Learning as basis-algorithm in the learning process. When the agent is inserted in small/discreet environments, the results through methods such as Q-learning are satisfactory (Sen & Sekaram, 1995; Claus & Boutilier, 1998). However, when the environment is large/continuous, the application of Reinforcement Learning methods using "lookup tables" becomes impracticable, due to the great dimension of the space of states. This problem is known as "curse of dimensionality" (Kaelbling et. al. 1996; Sutton & Barto, 1998, Ribeiro, 1999).

In MAS, such issue is considerably greater, since the necessary memory growth also becomes exponential to the quantity of agents involved in the application. The values must represent a function of the actions of all agents combined. In

order to go around it, some kind of generalization must be incorporated to the state representation. The generalization consists in producing modifications in the RL methods through function approximation, allowing the update through fortification not only of the state related to the current iteration, but also of other states that are correlated (Ribeiro, 1999; Sutton, 1996). Generalization has shown the way to the identification of similar states. However, the detailed ignorance towards the environment increases the difficulty to define how these states must be grouped. The function approximation can be done with many intelligent techniques (Ribeiro, 1999, Kaelbling et. al.,1996), such as Neural Networks and Fuzzy Logics (Sutton, 1996; Jouffe, 1998).

Some of the current models, which use function approximation, demand a great deal of information or previous definitions. As a whole, the limitations of the main RL-based and Fuzzy Logics models, such as NEFCON (Neuro-Fuzzy Controller) (Nauck e Kruse, 1997) and FQL (Fuzzy-Q-Learning) (Glorennec, 1997), are the need for predefinition of fuzzy set number and the pertinence function of such sets, used in the previous and the following of the fuzzy rules; the need for predefinition of the amount of rules and even of the predecessors that compose the rules; the number of entry variables in the model.

Aiming at overcoming such limitations, Hierarchic Reinforcement Learning Neuro-Fuzzy (RL-NFH) (Figueiredo, 2003) models were developed, contemplating the self-learning of a single agent. Originally, Souza (2002) created this family of Neuro-Fuzzy Systems, named Hierarchic Neuro-Fuzzy Systems, to achieve learning with supervised training in descent gradient. Due to the success achieved in the development of RL-NFH models of unsupervised fortification-based learning, this research has focused on developing new Multi-Agent Systems from this model family in large/continuous environments.

2 RLF-HNFP Model

The RL-HNFP model is composed of one or various standard cells called RL-neuro-fuzzy BSP (RL-NFP). These cells are laid out as a hierarchical structure in the form of a tree, according to the partition being used. The BSP – Binary Space Partitioning (Souza et al., 2002; Vellasco et al., 2004) divides the space successively, in a recursive way, in two regions. The Politree partitioning is a generalization of the quadtree (Souza et al., 2002). In this partitioning the subdivision of the n-dimensional space is accomplished in $m=2^n$ subdivision.

Figure 1a shows an example of a two dimensional input partitioned using the BSP method. The Politree partitioning can be represented by a tree structure. Figure 1b presents the generic politree partitioning (with n inputs).

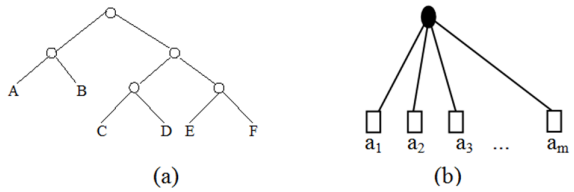

Figure 1 – (a) BSP Tree representing the BSP partitioning, (b) generic tree representation of the politree partitioning.

Hierarchical 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. The outputs of the cells in the lower levels are the consequents of the cells in the higher levels.

An RL-NFP cell is a mini-neuro-fuzzy system that performs n-dimensional partitioning of a given space in accordance with the membership functions. The RL-NFP cell generates a precise (crisp) output after the defuzzification process (Vellasco et al., 2003; Figueiredo et al., 2004). Each cell has only one input (x) and the value of this input variable is read by one of the agent's sensors. Then it is inferred in the antecedents' fuzzy sets (*low* - $\rho(x)$ - and *high* - $\mu(x)$).

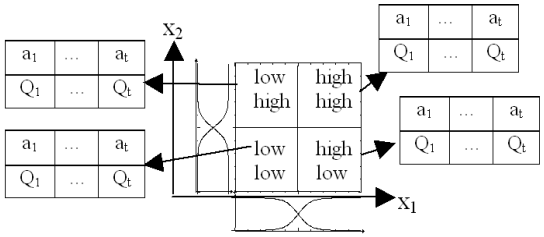

Figure 2 – Internal representation of the RL-NFP cells

The consequents of the cell's partitions may be of the singleton type (a constant) 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 partition has a set of possible actions (a_1, a_2, ... a_1), where 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 p. For further details about RL theory, see (Sutton and Barto, 1998).

The linguistic interpretation of the mapping implemented by the RL-NFP cell is given by rules. Each rule corresponds to one of the polipartitions generated by the BSP partitioning and has the following structure:

- Rule 1: If $x_1 \in \rho_1$ and $x2 \in \rho_2$ then $y = a_i$
- Rule 2: If $x_1 \in \rho_1$ and $x2 \in \mu_2$ then $y = a_j$
- Rule 3: If $x_1 \in \mu_1$ and $x2 \in \rho_2$ then $y = a_p$
- Rule 4: If $x_1 \in \mu_1$ and $x2 \in \mu_2$ then $y = a_q$

RL-HNFP models can be created based on the interconnection of the basic cells described above. The cells form a hierarchical structure that results in the rules that compose the agent's reasoning.

Neuro-fuzzy learning process is generally divided into two parts: structure identification and parameter adjustment (Jang, 1993). RL-HNFP performs these learning tasks in a single algorithm. It consists of six main steps that are: generate root cell, calculate global reinforcement, back-propagate the reinforcement, select actions, update Q-values and partitioning. For details about how each step works, see (Figueiredo et al., 2004).

3 Multi-Agent RLF-HNFP Model

The main idea of the multi-agent version of RL-NFH family algorithm is to make every single agent able to explore different tasks or different state-action pairs simultaneously, therefore speeding up the learning and the convergence for an optimal policy. There are many types of approach involving multi-agents, from the goal to be learned, coordination among agents and homogeneity of learning. Through the development of the Multi-Agent RLF-HNFP (MA-RL-HNFP) model proposal, it was intended to broaden the RL-HNFP model in different ways in order to contemplate the greatest possible variety of applications.

In this paper, different types of MA-RL-HNFP systems created from the original will be presented, describing their behavior, particularities and objectives. Besides that, the learning model differences, in coordination and learning dynamics, will also be explained. The proposed MA-RL-HNFP systems differ in two main aspects: learning dynamics and the mechanism of coordination among agents. And, as for the learning dynamics of multiple agents, it basically happens in two ways: with or without the sharing of the learning structure. When sharing the structure, the learning structure is the same and unique to every agent. It works as a "group intelligence", which is used by all agents for decision-making, and also as knowledge repository. In other words, each agent can execute a specific action in a distinct state of the problem and nurture the structure with the acquired knowledge. The following agents access the "deposited" information by the previous agents in the single structure, choose the action to be performed and update the Q values according to the received reinforce.

It is worth mentioning that this type of model, where multiple agents use the same structure, is indicated for issues that request the agents to learn a same task and seek cooperation. If they all compete using the exact same "intelligence", there would be no losers or winners. Besides that, in a competing environment, there is no point in an agent providing the obtained knowledge to the rivals.

In a similar way to the RL-HNFP model, learning also occurs cyclically. First, the agents access the structure to make a decision, explore the state-action pairs of the environment, receive the return and transfer "what they have learned" to the knowledge structure, which processes the information of the explored state-action pair and the received return value. Then, a new cycle is performed for another agent, which can explore a completely distinct space of the environment, in relation to the previous agent. The main change is the agent rotation in the exploring of the environment. The full training process is divided into seven steps, as depicted in Figure 3.

The other way of learning is when each agent keeps its own structure, apart from other remaining agents in the environment. In this case, each agent specializes itself to a certain different task. This approach aims at contemplating problems involving both cooperation and competition among multiple agents.

In order to make agents cooperate, there must be a coordination mechanism, or even a central agent, receiving and combining the information from each RL-HNFP structure. In competitive problems, the agents may use their own structures

independently, aiming at learning a specific task and competing against another agent, equally "equipped" with a RL-HNFP learning structure.

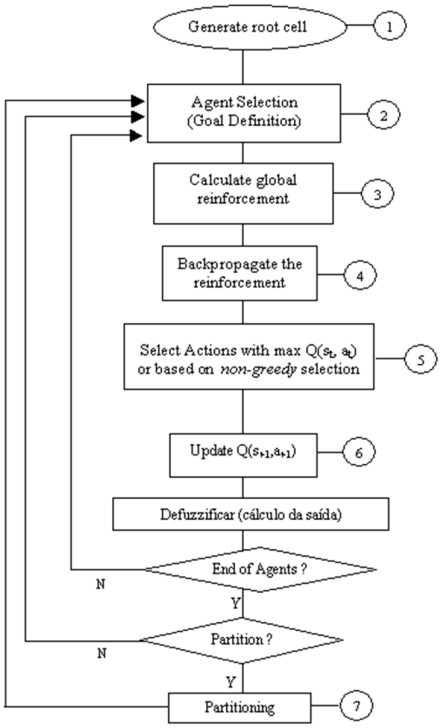

Figure 3 –Learning Algorithm of the MA-RL-HNFP model.

The agent individual goal coordination can happen in basically two ways:

- Explicit coordination: a coordination mechanism may be included, for example, to define what the selected agent's goal is in that step.
- Implicit coordination: there not being a global coordination mechanism, the second possibility is to make the agents redefine their strategies every turn, not communicating to the remaining, according to its perception of the environment.

The developed MA-HNFP models will be explored through the Pursuit game (Benda et al., 1986). In its most traditional way, 4 predators and 1 prey are part of it, in an orthogonal grid, divided by lines and columns, where each space in the grid cannot be occupied by two participants simultaneously. The goal of the game is to make the predators capture the prey as quickly as possible.

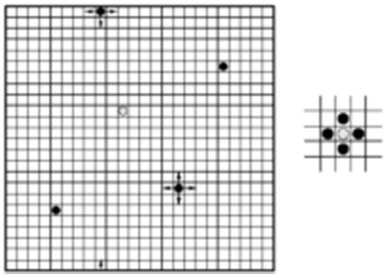

Figure 4 - The Pursuit Game (Benda et al., 1986).

4 Case Study

In this task, the MA-RL-HNFP systems for the Pursuit problem were built following the specifications, premises and restrictions below:

- space composed by an orthogonal 9x9 positions grid;
- the 4 predator agents move by using the MA-RL-HNFP, with the sole purpose of capturing the prey;
- prey and predators can only move in 4 directions. Diagonal movements are not allowed;
- each turn is composed of one move per participant;
- there is no acceleration concept. Each move is always performed to the right, to the left, above or below;
- each predator agent can visualize the positioning of the remaining agents and the prey's;
- the predator agents don't know the goals of the remaining agents;
- the agents can share the 'knowledge' that was obtained throughout the learning process;
- in case of collisions after a move, the agent or prey returns to their previous spot;
- if a participant tries to move towards the end of the grid, it also returns to the previous spot.

A characteristic that varied during the tests was the coordination among predator agents. At first, there wasn't a centralizing agent to perform the coordination among agents. Afterwards, a coordination mechanism was inserted to control the agent arriving goals in relation to the prey. As for the movements of the prey, it first moved randomly in every turn. Then, in a second configuration, the prey was also an intelligent agent. It acted aiming at running away and making the capture harder, simulating a competitive MAS. Taking into account the characteristics above, it's possible to say that the problem consists in a MAS application with a homogeneous learning dynamics, as all the agents use a same learning algorithm (even the prey when acting as an intelligent agent).

As for communication, it happens indirectly. When using the same MA-RL-HNFP structure, in a certain way, the agents are sharing the acquired knowledge. However, it's worth remembering that the agents decide independently where to get to in order to capture the prey (up, down, left, right), for the goals are not "public". The shared information in the tree only represent a mapping of the states and actions of the problem.

The state was defined with a single variant: the relative position of the prey in relation to the agent, represented by the arctangent angle formed by the prey, considering that the agent is always on the axis, or on the (0, 0) position. The value of the angle, which varied from 0º to 360º, was always normalized. For a better understanding, the picture below shows eight possible state values for the agent, when it is found in the (4, 4) position. For example, if the agent, represented below by the X in the gray cell, is in the (4, 4) position, and the prey in the (7, 7) cell, the formed arctangent angle is 45°. This value is sufficient to determine the state.

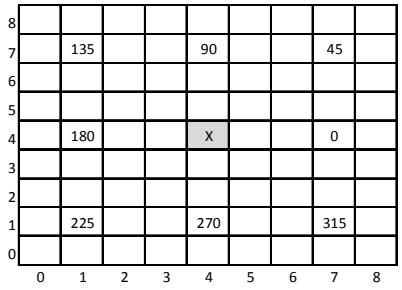

Figure 5 - Definition of eight possible state values.

The reinforcement received by the model after each action is inversely proportional to the agent's distance to the prey, as shown in the equations below.

Distance: $d = (A_x - P_x) + (A_y - P_y)$ (1)

Reinforcement: $r = 1 - d_{normalized}$ (2)

, where A_x and P_x are the agent's and the prey's positions on the x axis, and A_y and P_y are the agent's and the prey's positions on the y axis, respectively; and dnorm is the distance between the prey and the agent normalized in a linear way from 0 to 1.

The training process was performed with the prey always in the (4, 4) position. The position of each agent in the beginning of the game, or after each capture, was also initialized in a fixed way, always on the corners of the grid - (0, 0), (8, 8), (8, 0), and (0, 8) positions. In other words, the agents needed to leave the corners and capture a prey which was fixed exactly in the center of the grid.

For the capture to be performed, in each step, the agents define the "goals" they must reach. Considering the prey's (x, y) position as (P_x, P_y), the four possible goals are: (P_x - 1, P_y), (P_x + 1, P_y), (P_x, P_y - 1) and (P_x, P_y +1). As each agent can visualize both the prey's and the other agents' positions, the goal choice algorithm used by the agents has always tried to find the formation in which the adding of the distance among agents and goals would be as small as possible. Eventually, there can be goal collisions, as one agent doesn't know the others' goals. This kind of collision must be solved naturally throughout the moving of the agents, as the approach the prey and, consequently, their goals.

5 Results

The training was performed during 100 steps. Analyzing the training, the first capture only occurs in step 16. All the moves, correct or not, cooperate in enriching the knowledge of the MA-RL-HNFP structure, which is used in the moving decision and is updated as the agents receive recurrence of their actions.

The second capture is already performed in a great way, in only 7 steps. Obviously, not all following captures took only 7 steps, which is the smallest distance between the agent and the prey (the optimal path). It only happened because the selection method of the actions in the RL-HNFP algorithm, during the learning process, does not always choose the action that has the highest Q value. However, it's possible to picture the convergence speed of the method. In total, throughout the 100 training cycles, 12 captures were performed. The picture below shows the necessary quantity of steps to perform the captures.

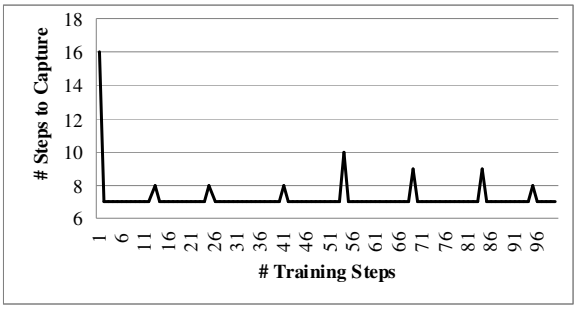

Figure 6 - Evolution of the number of steps needed for the first 100 captures.

To evaluate the convergence time gain of the multi-agent version in relation to the original one, a similar problem was built with only the prey and one predator agent. There was a training with 300 iterations, in which the prey also stood on the central position of the grid (4, 4) and a single agent started from one of the corners of the grid – positions (0, 0), (0, 8), (8, 0) and (8, 8). Every 75 steps, the agent started from a different corner of the grid. The goal was to get to the (4, 4) position, where the prey stood still. In this case, the smallest quantity of steps needed is 8, from any agent's starting position. The training results of the RL-HNFP original version can be visualized in the graph below. The picture shows that the convergence happened only after the 160th step of the training, which is a 10 times longer period than the one demanded by the multi-agent algorithm.

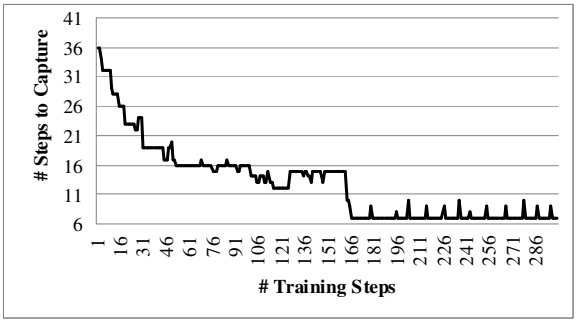

Figure 7 - Evolution of the number of steps needed for the single agent reach the prey.

In order to perform tests of the RL-HNFP model for multi-agents, new procedures were adopted. This time, the position of the prey was randomly initialized. The MA-RL-HNFP structure was used by the agents in every move-making, seeking the goal. In the first set of tests, a total of 1000 pursuits were performed, with the predators always starting from the corners of the grid. The average step count until the capture of the prey was 12.983. In other words, 12983 steps were necessary for the prey to be captured 1000 times. Then, another 1000 pursuits were performed. This time, the position of the predators in the beginning was also random. The average step count until the capture of the prey was 9.476, a little slower than the fixed-position start.

In the tests that were performed and mentioned above, each agent defined its objectives individually, based on its perception of the environment. The goal choice used by the agents always aimed at finding the formation in which the adding of the distance between agents and goals was as small as possible. So, throughout the pursuits, there were agents' goals collisions. In other words, as an agent didn't know the goals of the remaining ones, two or more agents could choose the same finish spot during a certain moment of the game. These collisions were solved naturally throughout the moving of the agents, as they approached the prey and, consequently, their goals.

New tests were performed after the introduction of a coordination mechanism among the agents. This time, the goals were no longer chosen by the agents, but defined by a central agent, which had a broad view of the environment. Then, goal collision stopped during the pursuits. The central agent, in every turn, chose different finish spots for the agents, according to each one's position.

At first, the tests were performed with the agents always starting from the four corners of the grid. After 1000 captures, the average step count until capture was 11.236, 13.45% smaller than the 12.983 reached when the game was performed without an explicit coordination among agents. Then, another 1000 pursuits were performed with the predators being initialized randomly. The average step count until capture was 7.695, 18.79% smaller than the 9.476 step average obtained without the coordination mechanism.

Table 1 - Comparisons between the performance of the different MA-RL-HNFP systems

Predator Start	Prey Start	No Coordination	With Coordination	Pursuit Time (%)
Random	Random	9.476	7.695	-18,8%
Fixed	Random	12.983	11.236	-13,5%
Random	Fixed	9.102	7.404	-18,7%
Fixed	Fixed	11.583	9.654	-16,7%

One last application was developed, this time, with the prey acting in an intelligent way. In other words, instead of moving randomly, the prey also started moving based on a RL-HNFP structure, after the learning process, competing against the predator agents. The results have shown that the introduction of the intelligence mechanism can prevent a great deal of captures, as shown in the table below.

Table 2 - Analisys of the performance after introducing an Intelligent Prey Agent using the RL-HNFP model

		Random Prey		Intelligent Prey	
Predator Start	**Prey Start**	**No Coordin.**	**With Coordin.**	**No Coordin.**	**With Coordin.**
Rand.	Rand.	9.476	7.695	21.323	18.437
Fixed	Rand.	12.983	11.236	18.978	16.175
Rand.	Fixed	9.102	7.404	19.812	17.367
Fixed	Fixed	11.583	9.654	N/D	278.323

6 Conclusions

The goal of this paper was to develop multiple-agent systems using reinforcement hybrid models of the RL-NFH family. The choice of the RL-NFH family models is due to their autonomous capacity to create rules and expand their own rule structure, extract knowledge from the direct interaction between the agent and the environment, without using supervised algorithms; and

to produce linguistically interpretable results, in the form of fuzzy rules. In the multi-agent version, the main idea is to make them able to explore the state-action space simultaneously and cooperate to the learning of the same structure, speeding up the learning of a certain task.

The developed models were explored through a case study of the Pursuit game. The tests performed by comparing the performance of the learning of a single agent with the performance of four agents learning in parallel brought up the great difference in the quantity of necessary iterations for the algorithm to reach optimal structure. Besides that, it was also possible to show the implicit coordination among agents during the capture of the prey, even though the agents could not access the goals, or "finish lines" in relation to the prey, of the remaining participants.

New tests were performed after the introduction of an explicit coordination mechanism, working as a "central agent", aiming at synchronizing the predator agents' goals in every turn. The results showed the reduction in the needed turns average until capture, making the profit from the creation of the mechanism clear. Finally, it was also simulated a competitive nature problem, in which the prey started acting in an intelligent way, using a RL-HNFP structure. The results showed that the introduction of the intelligence mechanism can avoid a great deal of captures.

7 References

[1] W. Brenner.; R. Zarnekow; Wittig, H. Intelligent Software Agents. Springer, 1998

[2] L. Busoniu; R. Babuska; B. De Schutter. A Comprehensive Survey of Multiagent Reinforcement Learning. In: Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions, Volume: 38, Issue: 2, Mar, 2008

[3] C. Claus; C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In: Proc. 15th Nat. Conf. Artif. Intell. 10th Conf. Innov. Appl. Artif. Intell. (AAAI/IAAI-98), Madison, WI, Jul. 26–30, pp. 746-752

[4] J. Clouse. Learning from an automated training agent. Presented at the Workshop Agents that Learn from Other Agents, 12th Int. Conf. Mach. Learn. (ICML-95), Tahoe City, CA, Jul. 9–12

[5] K. Figueiredo.; M.M.B.R. Vellasco; M.A.C. Pacheco. Novos Modelos Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço para Agentes Inteligentes, Tese de Doutorado, Departamento de Engenharia Elétrica Pontifícia Universidade Católica do Rio de Janeiro, Fev, 2003

[6] R. Fitch; B. Hengst; D. Suc; G. Calbert; J.B. Scholz. Structural abstraction experiments in reinforcement learning. In: Proc. 18th Aust. Joint Conf. Artif. Intell. (AI-05), LNCS, vol. 3809, Sydney, Australia, Dec. 5-9, pp. 164-175

[7] P. Y. Glorennec; L. Jouffe. Fuzzy Q-Learning. In: Proceedings of Fuzz-Ieee'97, Sixth International Conference on Fuzzy Systems, Barcelone, Espagne, Juillet 1997, p. 659-662

[8] N.R. Jenning; M.J. Wooldridge. Agent Technology: Fundations, Applications, and Markets. Springer, December, 1997.

[9] L. Jouffe. Fuzzy Inference System Learning by Reinforcement Methods. IEEE Transactions on Systems, Man and Cybernetics, Part C, vol. 28, n. 3, p. 338-355, 1998

[10] L.P. Kaelbling; M.L. Littman; W.A. Moore. Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4, May 1996, p. 237-285

[11] J.R. Kok; P. J.'T Hoen; P.B. Bakker; N. Vlassis. Utile coordination: Learning interdependencies among cooperative agents. In: Proc. IEEE Symp. Comput. Intell. Games (CIG-05), Colchester, U.K., Apr. 4–6, pp. 29-36

[12] B. Price; C. Boutilier. Accelerating reinforcement learning through implicit imitation. J. Artif. Intell. Res.,vol19, pp569-629, 2003

[13] C.H.C. Ribeiro. A Tutorial on Reinforcement Learning Techniques. In: International Joint Conference on Neural Networks ed.: INNS Press, 1999

[14] S. Sen; M. Sekaran. Multiagent coordination with learning classifier systems. In: Proceedings of of the IJCAI Workshop on Adaptation and Learning Multiagent Systems, Montreal, 1995, p. 84-89

[15] F. Souza, M.M.B.R. Vellasco, M.A.C. Pacheco. Hierarchical Neuro-Fuzzy QuadTree Models. Fuzzy Sets & Systems vol 130/2, 2002, 189-205

[16] R.S. Sutton;.A.G. Barto. Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press, 1998

[17] R.S. Sutton. 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

[18] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proc. 10th Int. Conf. Mach. Learn. (ICML-93), Amherst, OH, Jun. 27–29, pp. 330–337

[19] M.M.B.R. Vellasco, M.A.C. Pacheco, L.S. Ribeiro Neto; F.J. Souza. Electric load forecasting: evaluating the novel hierarchical neuro-fuzzy BSP model. Electrical Power and Energy Systems Elsevier Ltd 26, 2004, 131–142

[20] G. Weiss. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Edited by Gerhard Weiss, 1999