MODELO NEURO-FUZZY HIERÁRQUICO POLITREE COM APRENDIZADO POR REFORÇO PARA AGENTES INTELIGENTES

Karla Figueiredo[*]
karla@uerj.br

Marley Vellasco[†]
marley@ele.puc-rio.br

Marco Pacheco[†]
marco@ele.puc-rio.br

Flávio Souza[‡]
fsouza@uerj.br

[*]Departamento de Engenharia Eletrônica
Universidade Estadual do Rio de Janeiro
Rua São Francisco Xavier, 524 - Rio de Janeiro, 20550-900 RJ Brasil

[†]Departamento de Engenharia Elétrica
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225, Rio de Janeiro - 22453-900 RJ Brasil

[‡]Departamento de Engenharia de Sistemas e Computação
Universidade Estadual do Rio de Janeiro
Rua São Francisco Xavier, 524 - Rio de Janeiro, 20550-900 RJ Brasil

RESUMO

Este trabalho apresenta um novo modelo híbrido neuro-fuzzy para aprendizado automático de ações efetuadas por agentes. O objetivo do modelo é dotar um agente de inteligência, tornando-o capaz de, através da interação com o seu ambiente, adquirir e armazenar o conhecimento e raciocinar (inferir uma ação). Este novo modelo, denominado Reinforcement Learning Neuro-Fuzzy Hierárquico Politree (RL-NFHP), descende dos modelos neuro-fuzzy hierárquicos NFHB, os quais utilizam aprendizado supervisionado e particionamento BSP (Binary Space Partitioning) do espaço de entrada. Com o uso desse método hierárquico de particionamento, associado ao Reinforcement Learning, obteve-se uma nova classe de Sistemas Neuro-Fuzzy (SNF) que executam, além do aprendizado da estrutura, o aprendizado autônomo das ações a serem tomadas por um agente. Essas características representam um importante diferencial em relação aos sistemas de aprendizado de agentes inteligentes existentes. O modelo RL-NFHP foi testado em diferentes problemas benchmark e em uma aplicação de robótica (robô Khepera). Os resultados obtidos mostram o potencial do modelo proposto, que dispensa informações preliminares como número e formato das regras, e número de partições que o espaço de entrada deve possuir.

PALAVRAS-CHAVE: Agentes Inteligentes; Modelos Neuro-Fuzzy; Aprendizado por Reforço; Aprendizado Automático.

ABSTRACT

This work presents a new hybrid neuro-fuzzy model for the automatic learning of actions taken by agents. The objective of this model is to provide intelligence for an agent, making it capable of acquiring and retaining knowledge, as well as thinking (infer an action), by interacting with its environ-

ARTIGO CONVIDADO:
Versão completa e revisada de artigo apresentado no SBAI-2005
Artigo submetido em 01/06/2006
1a. Revisão em 28/08/2006
2a. Revisão em 07/03/2007
Aceito sob recomendação do Editor Convidado
Prof. Osvaldo Ronald Saavedra Mendez

ment. This new model, named Reinforcement Learning Hierarchical Neuro-Fuzzy Politree (RL-NFHP), descend from the hierarchical BSP neuro-fuzzy models, which employ supervised learning and BSP partitioning (Binary Space Partitioning) of the input space. By using this hierarchical partitioning method, together with the Reinforcement Learning methodology, a new class of Neuro-Fuzzy Systems (SNF) was obtained, which automatically learns its structure as well as the actions that must be taken by an agent. These characteristics represent an important differential when compared to existing intelligent agents learning systems. The RL-NFHP model was tested in different benchmark problems, as well as in a robotic application (Khepera robot). The results obtained demonstrate the potential of the proposed model, which does without information as number of rules, rules' format and number of partitions that the input space should have.

KEYWORDS: Intelligent Agents; Neuro-Fuzzy Models; Reinforcement Learning; Automatic Learning.

1 INTRODUÇÃO

Para um número cada vez maior de aplicações há a necessidade de que os sistemas computacionais tenham maior grau de autonomia, isto é, que incorporem a capacidade de autodecidir o que deve ser feito para satisfazer os objetivos determinados pelo programador. A autonomia é uma das características que permite classificar certos programas (códigos ou algoritmos) como agentes. No entanto, uma outra característica possui grande importância por aumentar a capacidade autônoma de um agente – a inteligência. Brenner et.al. (1998) são enfáticos ao afirmar que o agente deve ser capaz de aprender para ser considerado autônomo. O aprendizado, juntamente com a base de conhecimento e o raciocínio, integram a característica inteligência.

Sistemas como estes, cujo grau de complexidade é maior, têm sido concebidos através do uso de técnicas como Lógica Fuzzy (LF) (Mendel, 1995) e Redes Neurais (RN) (Haykin, 1998), e aplicados em áreas aonde a abordagem convencional não vinha conseguindo fornecer soluções satisfatórias. Muitos pesquisadores têm tentado integrar estas duas técnicas de modelagem de forma a gerar um modelo híbrido que possa associar as vantagens de cada abordagem e minimizar suas limitações e deficiências. Com este objetivo foram criados os sistemas híbridos neuro-fuzzy, ou simplesmente sistemas neuro-fuzzy.

Os modelos neuro-fuzzy tradicionais (ANFIS (Jang, 1993), NEFCLASS (Kruse, 1995) e FSOM (Vuorimaa, 1994)) ajustam apenas seus parâmetros ou têm uma capacidade limitada de ajuste em sua estrutura. Devido ao problema de explosão do número de regras, esses modelos são geralmente utilizados em aplicações com número restrito de entradas. Além disso, a maioria desses sistemas neuro-fuzzy tradicionais são adequados para treinamento supervisionado. Entretanto, quando a informação sobre a saída desejada não está disponível, é necessário realizar o aprendizado dos sistemas Neuro-Fuzzy através do algoritmo de Reinforcement Learning (RL). No entanto, quando o ambiente é grande e/ou contínuo, a aplicação de métodos de Reinforcement Learning tradicionais, utilizando lookup table (tabela de armazenamento das funções de valor para espaço de estados pequeno ou discreto) torna-se inviável, devido à grande dimensão do espaço de estados. Este problema é conhecido como *curse of dimensionality* (Barto, 2003; Lima, 2005; Ribeiro, 1999; Sutton, 1998). Para contorná-lo, alguma forma de generalização deve ser incorporada à representação de estados.

Essa generalização é geralmente realizada através da inserção nos métodos RL de técnicas de aproximação de funções, onde a atualização das funções de valor é efetuada através de reforços não apenas do estado relacionado à iteração atual, mas também de outros estados correlacionados por alguma característica comum (Ribeiro, 1999; Sutton, 1996; Moore, 1991). Os modelos atuais que utilizam aproximação de funções necessitam de um grande número de informações ou de definições prévias. De uma forma geral, os principais modelos baseados em RL e Lógica Fuzzy necessitam da prédefinição do número e formato dos conjuntos fuzzy usados nos antecedentes e conseqüentes das regras fuzzy, da prédefinição do número de regras e até mesmo dos antecedentes que compõem as regras, além da limitação quanto ao número de variáveis de entrada no modelo.

O modelo proposto neste trabalho foi concebido a partir do estudo das limitações nos modelos existentes e das características desejáveis para sistemas de aprendizado baseados em RL, em particular quando aplicados a ambientes contínuos e/ou ambientes considerados de alta dimensão (Moore, 1991; Boyan e Moore, 1995; Jouffe, 1998; Sutton e Barto, 1998).

Uma forma de minimizar o problema da dimensionalidade é o uso de particionamentos hierárquicos e recursivos (Souza 2002; Gonçalves, 2006). Através da preservação da independência das variáveis de entrada, esses particionamentos mantêm a interpretabilidade das regras fuzzy. O uso de métodos de particionamento recursivo em modelos neuro-fuzzy foi introduzido por Souza, obtendo excelentes resultados (Souza, 2000; Souza 2002; Gonçalves, 2006). Estes modelos de particionamento são flexíveis e minimizam o problema do crescimento exponencial do número de regras. Sua vantagem principal é permitir a criação de sistemas que constroem sua própria estrutura de forma automática.

Com o uso desse método de particionamento, associado ao Reinforcement Learning, obteve-se uma nova classe de Sistemas Neuro-Fuzzy (SNF), denominada Reinforcement

Learning-Neuro-Fuzzy Hierárquico (RL-NFH) (Figueiredo, 2005a).

O modelo RL-NFHP apresenta, portanto, as seguintes características: aprendizado automático da estrutura do modelo; auto-ajuste dos parâmetros associados à estrutura; capacidade de aprender a ação a ser tomada quando o agente está em um determinado estado do ambiente; possibilidade de lidar com um número maior de entradas do que os sistemas neuro-fuzzy tradicionais; e geração automática de regras lingüísticas com hierarquia.

O restante deste trabalho está organizado em mais 3 seções: a Seção 2 introduz o modelo RL-NFHP, descrevendo sua célula básica, sua arquitetura e seu método de aprendizado. A Seção 3 apresenta os resultados obtidos com quatro estudos de caso: 3 aplicações benchmark - carro na montanha (car-mountain problem), estacionamento do carro (cart-centering) e pêndulo invertido - e navegação do robô Khepera. Estas aplicações possuem número de variáveis de entradas e graus de complexidade diferentes que permitem comparar o desempenho do modelo proposto com outros modelos baseados em RL. Finalmente, a Seção 4 apresenta as conclusões deste trabalho.

2 REINFORCEMENT LEARNING - NEURO-FUZZY HIERÁRQUICO POLITREE

O modelo RL-NFHP tem três componentes fundamentais: o aprendizado por reforço, a característica fuzzy e o particionamento hierárquico.

No modelo RL-NFHP proposto, o aprendizado por reforço baseia-se no método SARSA (Sutton, 1998), conforme descrito na seção 2.3, sendo que o processo de identificação dos estados é realizado ao longo do aprendizado, não sendo conhecido previamente. As regras são geradas por um processo automático de particionamento do espaço de entrada. O componente RL faz com que o modelo aprenda a encontrar a ação mais adequada a ser executada para um determinado estado. O componente fuzzy do modelo agrega estados que têm comportamentos similares, associando-os a uma mesma ação. O aspecto hierárquico deste modelo refere-se ao fato de que cada partição do espaço de entrada define um subsistema que, por sua vez, pode ter como conseqüente um subsistema com a mesma estrutura (recursividade). Esta característica suaviza o processo de generalização das funções de valor (a árvore fuzzy hierárquica como aproximadora de função), não comprometendo os resultados, o que é um bom requisito para a convergência do processo (Sutton, 1996).

O modelo RL-NFHP é composto de uma ou várias células padrão chamadas células RL-neuro-fuzzy politree (RL-NFP). Estas células são dispostas numa estrutura hierárquica em forma de árvore. A célula de maior hierarquia gera a saída. As de menor hierarquia trabalham como conseqüentes das células de maior hierarquia. Nas próximas seções serão descritos, a célula básica, a estrutura hierárquica e o algoritmo de aprendizado.

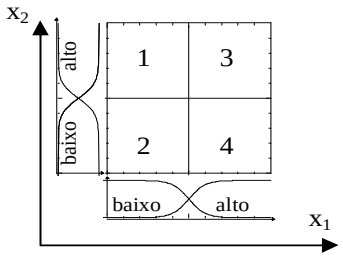


Figura 1: Funções de pertinência da célula RL-NFP com duas entradas

2.1 Célula Básica RL Neuro-Fuzzy Politree

Uma célula RL-neuro-fuzzy politree (RL-NFP) é um mini-sistema neuro-fuzzy que realiza um particionamento politree em um determinado espaço, utilizando, para cada variável de entrada, as funções de pertinência descritas na Figura 1 e equações 1 e 2.

A célula RL-NFP gera uma saída exata (crisp) após um processo de defuzzificação, conforme será mostrado posteriormente. Apenas para efeito de ilustração, na representação da célula (Figura 2) serão utilizadas apenas duas entradas, de forma a tornar as figuras mais simples do que com a forma n-dimensional proposta para o Politree.

A Figura 2 apresenta o interior da célula RL-NFP. Nesta figura as entradas x_1 e x_2 geram os antecedentes das quatro regras fuzzy após serem computados os graus de pertinência $\rho_x(x_1)$, $\mu_x(x_1)$, $\rho_y(x_2)$ e $\mu_y(x_2)$, onde: ρ_ι e μ_ι são os conjuntos fuzzy baixo e alto relacionados à variável x_ι, respectivamente. Os valores definidos como conseqüentes são conjuntos de ações (a_1, a_2, ... a_t), onde cada ação está associada a uma função de valor-Q (Sutton, 1998). Através do aprendizado por reforço, uma ação de cada polipartição será definida como aquela que representa o comportamento desejado do sistema quando o mesmo se encontra em um determinado estado. Um estado é definido por um conjunto de células ativas ao mesmo tempo.

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

$$\rho(x) = 1 - \mu(x) \tag{2}$$

Cada célula avalia todas as variáveis de entrada, formando os antecedentes das regras. Já os conseqüentes das polipartições da célula podem ser do tipo singleton (inferência Takagi-Sugeno de ordem zero) ou ser a saída de um estágio de nível anterior na arquitetura do modelo RL-NFHP. Apesar do conseqüente singleton ser simples, este não é conhecido previamente. Os valores definidos como conseqüentes singleton são ações que pertencem a um conjuntos de ações possíveis (a_1, a_2, ... a_t), como mostrado na Figura 2.

Estas funções de pertinência possuem dois parâmetros, 'a' e 'b', que definem os perfis das funções alto (μ) e baixo (ρ) de cada variável de entrada. Os α_i (ver Figura 2) simbolizam os níveis de disparo das regras. Estes níveis de disparo são calculados usando-se uma operação AND (T-norm) sobre os graus de pertinência de ρ_1, μ_1, ρ_2 e μ_2, conforme descrito a seguir:

$$\begin{aligned} \alpha_1 &= \rho_1(x_1) * \rho_2(x_2); \\ \alpha_2 &= \rho_1(x_1) * \mu_2(x_2); \\ \alpha_3 &= \mu_1(x_1) * \rho_2(x_2); \\ \alpha_4 &= \mu_1(x_1) * \mu_2(x_2). \end{aligned} \tag{3}$$

onde o símbolo '*' representa a operação AND, que pode ser realizada pela multiplicação ou pela operação de mínimo entre os dois valores.

A interpretação do conhecimento armazenado na célula RL-NFP da Figura 2 é representada pelo seguinte conjunto de regras lingüísticas:

regra1: Se $x_1 \in \rho_1$ e $x_2 \in \rho_2$ então y = a_i

regra2: Se $x_1 \in \rho_1$e $x_2 \in \mu_2$ então y = a_j

regra3: Se $x_1 \in \mu_1$ e $x_2 \in \rho_2$ então y = a_p

regra4: Se $x_1 \in \mu_1$ e $x_2 \in \mu_2$ então y = a_q

Cada regra corresponde a um quadrante da Figura 1. Quando os valores das entradas incidem sobre o quadrante 1, é a regra 1 que tem maior nível de disparo. Cada quadrante, por sua vez, pode ser subdividido em quatro partes através de uma outra célula RL-NFP. É muito importante lembrar que os conseqüentes a_i não são valores predeterminados, eles fazem parte de um conjunto de ações que deve ser explorado para que se possa determinar, através do aprendizado por reforço, a ação mais adequada para cada regra.

A saída 'y' de cada célula RL-NFP é dada pela média ponderada mostrada na equação (4) abaixo.

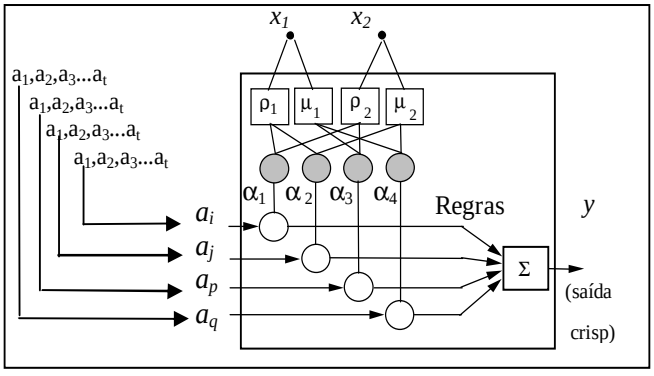


Figura 2: Interior da célula Reinforcement Learning Neuro-Fuzzy Quadtree (Politree com n=2)

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

onde n é o número de entradas, α_i é o grau de ativação (nível de disparo) da regra i e a_i corresponde a um dos dois conseqüentes possíveis abaixo:

- um singleton (conseqüente fuzzy singleton, ou Sugeno de ordem zero), caso em que a_i= constante;
- saída de um estágio de nível anterior, caso em que a_i = y_j, onde y_j representa a saída de uma célula genérica 'j', cujo valor é calculado, também, pela equação (4).

Como o valor do conseqüente singleton (valor da ação) não é conhecido previamente, utiliza-se o algoritmo de Reinforcement Learning para determinar a melhor ação para cada regra.

De acordo com as equações (1) e (2), as funções de pertinência utilizadas na célula básica RL-NFP são complementares. Isso leva a uma simplificação no procedimento de defuzzificação, uma vez que o somatório do denominador da equação (4), é igual a 1 para quaisquer valores de entrada x_i.

Desta forma, a saída da célula básica pode ser simplificada, como mostra a equação (5), a seguir.

$$y = \sum_{i=1}^{2^n} \alpha_i . a_i \tag{5}$$

As células RL-NFP formam uma estrutura hierárquica que resulta nas regras que compõem o raciocínio do agente. Para este modelo, os antecedentes das regras são definidos pelas variáveis de entrada, as quais estão associados dois conjuntos fuzzy. No caso deste modelo, todas as variáveis de entrada do sistema compõem os antecedentes das células. Os valores das variáveis de entrada são lidos pelos sensores do agente e são avaliados nos conjuntos fuzzy dos antecedentes. Se o antecedente é verdadeiro (resultado da aplicação do operador T-norm é diferente de zero), a regra é disparada. Os conseqüentes são as ações que o agente deve aprender ao longo do processo e são realizadas pelo seu atuador. Sendo assim, o modelo RL-NFHP também cria e determina sua estrutura mapeando estados em ações.

2.2 Arquitetura RL-HNFP

A arquitetura RL-NFHP é formada pela interligação de células básicas em uma estrutura em árvore.

Para cada variável de entrada existem duas partições definidas pelos conjuntos fuzzy: baixo e alto (Figura 1). Portanto, o número total de partições é dado pelo número de combinações entre os conjuntos baixo e alto de cada entrada. Deste modo, a Figura 1 apresenta o particionamento com duas entradas (22 = 4 partições).

Na árvore da Figura 3 os nós com pequenos círculos representam regiões que foram subdivididas. Os nós simbolizados por pequenos quadrados são nós terminais e representam as polipartições, isto é, as regiões que não sofreram subdivisões. A raiz da árvore simboliza todo o espaço a ser particionado. No exemplo da Figura 3 as polipartições i, onde $i \neq 2$ e $i \neq m$, não foram subdivididas, portanto os conseqüentes de suas respectivas regras são os valores a_i. As partições 2 e m foram subdivididas e os conseqüentes de suas regras são as saídas (y_2 e y_m) dos subsistemas 2 e m. Estes, por sua vez, têm como conseqüentes os valores $a_{21}, a_{22}, ..., a_{2m}$, e $a_{m1}, a_{m2}, ..., a_{mm}$, respectivamente. Cada 'a_i' corresponde a um conseqüente de Sugeno de ordem 0 (singleton), representando a ação que será identificada (dentre as ações possíveis), através de aprendizado por reforço, como sendo a mais favorável para um determinado estado do ambiente.

A saída do sistema exemplificado na Figura 3 é dada pela equação (6).

$$y = \alpha_1.a_1 + \alpha_2 \sum_{i=1}^{2^n} \alpha_{2i}.a_{2i} + \alpha_3.a_3 + \alpha_4.a_4 + \ldots + \\ + \alpha_m \sum_{i=1}^{2^n} \alpha_{mi}.a_{mi} \quad (6)$$

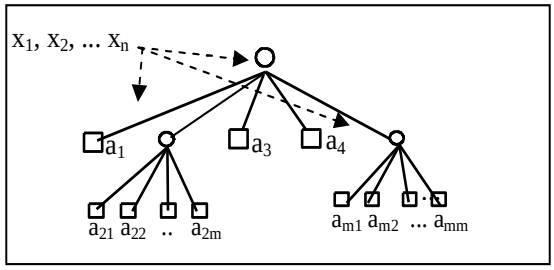


Figura 3: Árvore Politree

De uma forma genérica, a equação de saída de um sistema RL-NFHP de dois níveis completos é dada pela equação (7) abaixo. Neste caso houve necessidade de se incluir as variáveis k_i e k_{ij}. Essas variáveis assumem apenas valores iguais a '0' ou '1', indicando a existência ou não das polipartições de ordem 'i' e 'ij', respectivamente.

$$y = \sum_{i=1}^{2^n} \alpha_i k_i a_i + \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \alpha_i \alpha_{ij} k_{ij} a_{ij} \quad (7)$$

Expandindo a equação (7) para um sistema RL-NFHP de quatro níveis de hierarquia tem-se a seguinte fórmula:

$$y = \sum_{i=1}^{2^n} \alpha_i k_i a_i + \\ \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \alpha_i \alpha_{ij} k_{ij} a_{ij} + \\ \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \sum_{p=1}^{2^n} \alpha_i \alpha_{ij} \alpha_{ijp} k_{ijp} a_{ijp} + \\ \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \sum_{p=1}^{2^n} \sum_{q=1}^{2^n} \alpha_i \alpha_{ij} \alpha_{ijp} \alpha_{ijpq} k_{ijpq} a_{ijpq} \quad (8)$$

onde :

- α_i, α_{ij}, α_{ijp}, α_{ijpq}, são os níveis de disparo das regras de cada polipartição i, ij, ijp, ou ijpq, respectivamente;
- k_i (k_{ij}, k_{ijp}, k_{ijpq}), é igual a "1" se a partição i (ou ij, ou ijp ou ijpq) existe e "0" caso contrário;
- ' a_i, a_{ij}, a_{ijp}, a_{ijpq}, são os conseqüentes (*singletons*) das regras existentes.

Nas equações (6) a (8) já se levou em consideração a simplificação causada pelo uso das funções de pertinência comple-

mentares ($\rho+\mu=1$) no método de defuzzificação das saídas de cada subsistema neuro-fuzzy.

Os antecedentes das regras do modelo RL-NFHP são gerados a partir da leitura do ambiente, feita pelo agente através de seus sensores (s_1, s_2, ..., s_n). Os valores desses sensores podem ser utilizados diretamente ou então traduzidos em uma ou mais variáveis de entrada (x_1, $x_{2,...}$ x_n) das células. Todas as variáveis x_i são consideradas no momento de criação de cada célula da hierarquia e, dependendo de seus valores, a célula pode se tornar ativa ou não. A ativação de uma célula significa que o agente está em um estado definido pelo domínio dos conjuntos fuzzy do antecedente da regra (domínio da entrada da célula).

Conforme exemplificado no conjunto de regras apresentadas na seção 2.2, os conseqüentes das regras fuzzy podem ser de dois tipos: Conseqüente Tipo 1 – Singleton, se a célula for uma folha da estrutura; e Conseqüente Tipo 2 – Célula RL- NFP, se o antecedente (e sua partição) estiver associado a uma célula intermediária.

No caso de Conseqüente Tipo 1 –Singleton, as regras fuzzy são do tipo apresentado na seção 2.1.

Os conseqüentes deste tipo são as ações que devem ser identificadas através do aprendizado por reforço, cujo objetivo é identificar as ações associadas às partições definidas pelas combinações entre os conjuntos fuzzy baixo e alto da célula RL-NFP que melhor respondam ao estado atual do agente. Conforme detalhado na seção 2.3 a seguir, o algoritmo de aprendizado por reforço utilizado no modelo RL-NFHP é baseado no SARSA (Sutton, 1998).

A próxima seção descreve o algoritmo de aprendizado desenvolvido para determinar tanto a estrutura do modelo quanto as ações a serem realizadas em cada polipartição.

2.3 Algoritmo de Aprendizado RL Neuro-Fuzzy Hierárquico Politree

O aprendizado em modelos neuro-fuzzy é geralmente dividido em duas etapas: a identificação da estrutura e, em seguida, o ajuste dos parâmetros. O modelo RL-NFHP realiza essas duas tarefas de aprendizado de forma integrada, em um único algoritmo. O fluxograma exibido na Figura 4 descreve o algoritmo de aprendizado do modelo RL-NFHP, onde se destacam seis passos correspondentes à numeração no fluxograma, os quais são descritos a seguir. O processo de aprendizagem começa com a definição das entradas relevantes para o sistema/ambiente no qual o agente está inserido e dos conjuntos de ações de que ele pode dispor para atingir seus objetivos. Os valores-Q iniciais associados às ações também devem ser definidos previamente. Normalmente, as

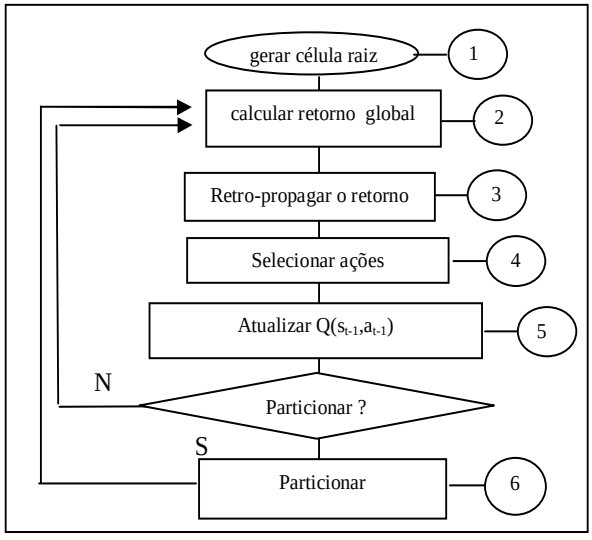


Figura 4: Algoritmo de Aprendizado do modelo RL-NFHP

aplicações que utilizam SARSA ou Q-Learning iniciam seus valores-Q com zero (Sutton, 1998).

O agente deve executar muitos ciclos para garantir o aprendizado do sistema/ambiente no qual o agente está inserido. Um ciclo é definido por um número de passos que o agente executa no ambiente, que vai do ponto em que ele for inicializado até o ponto considerado como sendo o seu objetivo. Cada passo compreende a execução do algoritmo que vai desde a leitura do ambiente até a execução da ação pelo agente.

1. Gerar célula raiz (ou célula pai)

   Uma célula raiz é criada tendo como domínio dos seus conjuntos fuzzy a faixa de valores das variáveis de entrada. Os valores das variáveis de entrada são normalizados com o objetivo de tornar o sistema mais genérico. Os valores correspondentes às variáveis de entrada da célula são lidos do ambiente e normalizados. Estes valores são avaliados nos conjuntos fuzzy baixo e alto, resultando em dois graus de pertinência, $\rho(x_1)\mu(x_1)$, ... $\rho(x_n)\mu(x_n)$, respectivamente para cada variável de entrada. Cada uma das polipartições escolhe uma das ações de seu conjunto de ações disponível, baseando-se nos métodos descritos no passo 4 do algoritmo – Seleção das ações. A saída da célula é calculada pelo processo de defuzzificação, dado pela equação (5), e representa a ação que será executada pelos atuadores do agente.

2. Retorno Global:

   Após a execução da ação, uma nova leitura do ambiente é realizada. Esta leitura permite que seja calculado

o valor de reforço do ambiente e se avalie a ação tomada pelo agente. Este valor deve ser calculado através de uma função de avaliação definida segundo os objetivos do agente. Assim, esta função de avaliação não define apenas a natureza do reforço, premiando ou punindo (+1/-1), mas também sua intensidade, tornando mais eficiente o processo de orientação deste agente durante o aprendizado. Exemplos de função de avaliação são apresentados na seção 3 - Estudo de Casos.

3. Retropropagar o retorno:

   A cada passo, no processo de aprendizado, o retorno é determinado para cada partição de todas as células ativas, mediante o cálculo de sua participação na ação resultante. Dessa forma, o retorno do ambiente é retropropagado a partir da célula raiz até as células folhas. Maiores detalhes ver (Figueiredo, 2004; Figueiredo, 2005b).

4. Seleção das ações

   As ações são associadas a funções de valores-Q e compõem um conjunto de ações que são selecionadas e experimentadas durante o aprendizado por reforço. A exploração do espaço de estados é fundamental à descoberta de ações que correspondam à melhor resposta do agente (que visa atingir um objetivo) quando este se encontra em um determinado estado do ambiente. Sendo assim, foi utilizado o método denominado $\in$-greedy (Sutton, 1998), que seleciona a ação associada ao maior valor esperado de Q com probabilidade (1-$\in$); e, com probabilidade $\in$, seleciona aleatoriamente uma ação qualquer. O valor máximo de $\in$ é 0.1 (10%) (Sutton, 1998).

5. Atualizar Q(s_{t-1},a_{t-1}):

   A partir dos valores de retornos calculados para cada célula da estrutura, os valores-Q associados às ações que contribuíram para a ação resultante executada pelo agente devem ser atualizados. Esta atualização é feita em função das comparações entre os retornos globais atual e anterior. A atualização dos valores Q ocorre de duas formas distintas: prêmio e punição. No caso do valor do retorno global atual ser maior que o retorno global anterior (R_{t+1}> R_t), o objetivo é premiar atualizando o valor de Q segundo a equação (9).

$$Q(s_t, a_t) = (1 - \alpha_t).Q(s_t, a_t) + \\ + \alpha_t[r_{t+1} + \gamma Q(s_{t+1,} a_{t+1})] \quad (9)$$

onde: o valor $Q(s_t, a_t)$(das células ativas em t) é atualizado a partir do seu valor atual $Q(s_t, a_t)$; r_{t+1} é o retorno local imediato (este é o retorno que é definido no passo da retropropagação do retorno global); γ é um parâmetro que fixa um percentual da contribuição do valor Q associado à próxima ação a_{t+1}escolhida ($Q(s_{t+1,} a_{t+1},)$) quando o sistema está no es estados_{t+1}; e α_t é o parâmetro proporcional à contribuição relativa desta ação local na ação global. O valor Q($s_{t+1,} a_{t+1}$) é calculado a partir da soma ponderada dos valores-Q (das células ativas em t+1) com relação aos graus de pertinência da variável de entrada da célula ($\rho Q_{baixo} + \mu$ Q_{alto}). Além da atualização de Q, neste caso (R_{t+1}> R_t) são reduzidas as taxas de *exploration/exploitation* $\in$-greedy que estão associadas a cada polipartição (conjunto de ações) de cada célula ativa.

No caso do retorno global atual ser menor ou igual ao retorno global anterior ($R_t \geq R_{t+1}$), o objetivo é punir atualizando o valor de Q segundo a equação (15).

$$Q(a_t, s_t) = (1 - fp).Q(a_t, s_t) \quad (10)$$

onde fp é o fator de punição que varia entre [0,1] e é definido como a relação entre o retorno local da partição e o retorno global. Quando o retorno global atual é menor que o retorno global anterior a atualização do valor Q (equação 10) tem por objetivo reduzir o valor Q, de modo a reduzir as chances de escolha desta ação quando esta célula estiver novamente ativa. Neste caso os parâmetros $\in$-greedy das partições envolvidas terão suas taxas aumentadas, permitindo que, nas próximas vezes que esta polipartição estiver ativa, outras ações diferentes tenham mais chances de serem escolhidas.

6. Particionar: Para a realização do particionamento cada bipartição deve atender a dois critérios. O primeiro critério evita que a estrutura cresça devido ao mau desempenho ocasionado pela escolha ainda imatura das ações, e o segundo critério incentiva o particionamento quando existe significantes variações nas funções de valor das ações.

   Quando uma bipartição possui todos os requisitos necessários para o particionamento, uma célula filha é criada e conectada àquela bipartição. Seu domínio será o subdomínio correspondente à bipartição baixa ou alta de seu ancestral mais próximo. As células filhas também herdam deste ancestral o conjunto de ações com seus respectivos valores Q.

   Quando uma polipartição possui todos os requisitos necessários para o particionamento, uma célula filha é criada e conectada àquela polipartição. Seu domínio será o subdomínio correspondente à polipartição de seu ancestral mais próximo. As células filhas também herdam

deste ancestral o conjunto de ações com seus respectivos valores Q.

Quando uma célula possui todas as células descendentes (todas as polipartições foram sub-divididas), todas as ações da célula ancestral tornam-se as ações de saída das células filhas e, neste caso, a célula ancestral tem suas funções de pertinência anuladas; ou seja, o grau de pertinência ρ e μ da célula ancestral passam a valer 1. O procedimento de desativar uma célula ancestral que já possua todos os seus descendentes contorna o problema de se ter na saída da estrutura valores de ações muito pequenos (comparados aos valores das ações nas polipartições) devido às múltiplas camadas da estrutura, e não prejudica o mapeamento (estado-ação), já que o domínio da célula pai já está representado nas células filhas com um grau de precisão maior.

Na próxima seção são apresentados os estudo de casos.

3 ESTUDO DE CASOS

Os estudos de casos foram escolhidos tendo como objetivo mostrar que o modelo RL-NFHP realiza aprendizado por reforço de ações efetuadas em ambientes grandes ou contínuos, sem conhecimento da dinâmica do ambiente, e com um mínimo de definições prévias. Conforme mencionado, estas características visam ampliar os limites da autonomia de um agente superando algumas limitações dos sistemas neuro-fuzzy existentes.

Assim, o modelo RL-NFHP foi avaliado em 3 problemas benchmark: Carro na Montanha, Cart-Centering e Pêndulo Invertido. Além disso, o modelo foi testado em uma aplicação de robótica, utilizando um simulador desenvolvido baseado no robô Kephera. Os resultados foram comparados com os modelos: CMAC Q-learning, Neural-Q-learning, FQL e FACL (Jouffe, 1998).

As sub-seções a seguir detalham a modelagem para cada um desses casos de estudo.

3.1 Carro na Montanha

O carro na montanha é um benchmark altamente relevante, uma vez que vem sendo usado por diferentes pesquisadores (Moore, 1991; Boyan, 1995; Jouffe, 1998, Sutton, 1998) com o objetivo de testar seus modelos de aproximação de funções.

O problema pode ser descrito da seguinte forma: um carro deve conseguir subir ao topo de uma montanha; entretanto, o carro possui potência menor do que a necessária para vencer a força da gravidade. Dessa forma, para alcançar seu objetivo, o carro precisa inicialmente se mover no sentido contrário ao alvo para poder adicionar a aceleração da gravidade à sua própria aceleração.

O problema possui duas variáveis de estados contínuas, a posição do carro $x_t \in [-1.2, 0.5]$ e sua velocidade $v_t \in [-0.07, 0.07]$, e três ações discretas: uma impulsão para a esquerda $(F = -1)$; nenhuma impulsão $(F = 0)$; e impulsão para a direita $(F = 1)$. A dinâmica do sistema é descrita pela equação (11).

$$\begin{aligned} v_{t+1} &= \min(0.07, \max(-0.07, v_t + 0.001F_t - \\ &\qquad -0.0025\cos(3x_t))) \\ x_{t+1} &= \min(0.5, \max(-1.2, x_t + v_{t+1})) \end{aligned} \tag{11}$$

Os valores de retorno foram calculados como funções de avaliação do ambiente. Estas avaliações levaram em conta a distância entre o carro e o objetivo (alto da montanha) e a velocidade do carro. O valor da função de avaliação referente à distância cresce à medida que o carrinho se aproxima do objetivo (equação (12)) ou quando ele se afasta do objetivo (equação (13)). Em ambos os casos, a função de avaliação relativa à velocidade cresce quando o valor do módulo da velocidade também cresce.

Se $(x_t < x_{t+1})$, então

$$R = k_1 e^{(-distância\_objetivo)} + k_2 e^{(|velocidade|)} \tag{12}$$

Se $(x_t > x_{t+1})$, então

$$R = k_1 e^{-(1-distância\_objetivo)} + k_2 e^{(|velocidade|)} \tag{13}$$

Os parâmetros k_1 e k_2 são constantes maiores do que 1 utilizadas para adequar os valores de retorno ao modelo.

O valor definido para o parâmetro γ da equação de atualização dos valores de Q foi de 1.0, a política *non-greedy* usada foi a de selecionar aleatoriamente uma ação e o valor inicial do parâmetro $\in$ (da política $\in$-greedy) foi estabelecido em 0.1 no momento de criação da célula. A taxa de incremento/decremento deste parâmetro quando a ação era respectivamente de punição ou prêmio, foi de 5%. Este valor foi definido heuristicamente, porém seguindo as recomendações feitas por Sutton (1996). Vale a pena lembrar que este valor representa a probabilidade de se realizar a seleção da ação utilizando-se métodos *non-greedy* (por exemplo, escolha aleatória da ação) e (1-$\in$) representa a probabilidade de se usar a ação que tiver associada ao maior Q. Estes são valores

típicos utilizados por outros pesquisadores (Sutton, 1998). Não foi usado o parâmetro *alfa-cut* neste experimento.

Foram realizados testes com variações das condições iniciais (posição e velocidade) e considerando diferentes conjuntos de ações. O objetivo destes testes foi avaliar o desempenho dos modelos em diversas situações.

A Tabela 1, a seguir, resume os resultados alcançados, na fase de aprendizado e na fase de teste, para diferentes tipos de configurações. Cada linha da tabela é a média dos resultados obtidos em 5 execuções para cada configuração.

Na Tabela 1, a primeira coluna indica o número da configuração utilizada para este modelo. As colunas posição/velocidade indicam as duas condições iniciais para o aprendizado: na primeira o carro, a cada ciclo, parte alternadamente de x = -0.5 (posição no "vale" da montanha) ou x = -1.2 (posição no extremo oposto ao objetivo a ser alcançado pelo carro) com velocidade zero; na segunda o carro parte aleatoriamente, a cada ciclo, de qualquer posição pertencente ao intervalo [-1.2, 0.5], com qualquer velocidade entre [-0.07, 0.07].

Na quarta coluna é indicado o conjunto de ações utilizadas em cada teste. A definição destes conjuntos foi feita de três formas diferentes: o conjunto A1 de ações é igual a {-1,0,1}, e a ação executada pelo agente é exatamente o valor da ação calculado na saída da estrutura pelo processo de defuzzificação; com A2 o conjunto de ações é o mesmo utilizado em A1, ({-1,0,1}), porém o valor da ação resultante após o processo de defuzzificação é submetido a uma função limitadora. Essa função tem por objetivo tornar o valor de saída discreto nos seguintes casos: se o valor na saída da estrutura estiver entre 0.1 e 1, o valor aplicado à equação (11) é 1; se estiver entre -0.1 e -1 o valor aplicado é -1, e se o valor estiver entre -0.1 e 0.1 utiliza-se o valor calculado na defuzzificação. Com A3, as ações disponíveis para cada célula pertencem ao conjunto {-10, -5, -1, 0, 1, 5, 10}; após a defuzzificação o valor de saída da estrutura têm os seus limites superior e inferior fixados em 1 e -1, respectivamente. Os testes com as condições A2 e A3 tiveram por objetivo permitir a comparação com modelos existentes, que têm por característica saídas discretas.

A coluna número de ciclos apresenta o número de ciclos estabelecido para efetuar o aprendizado, isto é, para o carro alcançar seu objetivo. A coluna tamanho da estrutura/número de parâmetros exibe a média do número de células ao final de 10 experimentos e o número de parâmetros (parâmetros a e b que definem os conjuntos fuzzy). Como cada célula tem 2 parâmetros, o número total de parâmetros será sempre igual ao dobro do número de células. A média de passos durante o aprendizado é o valor da média do número de passos considerando os 10 experimentos. A última coluna mostra os valores obtidos na fase de teste. Estes valores são a média do número de passos em 1000 experimentos com o carro partindo de qualquer posição $\in$ [-1.2,0.5] e velocidade $\in$ [-0.07,0.07].

Conforme se observa da Tabela 1, o modelo RL-NFHP apresentou uma boa capacidade de aprendizado; mesmo não partindo de pontos aleatórios do espaço de estados, o agente foi capaz de generalizar o conhecimento adquirido, devido à capacidade de generalização fornecida pelo componente fuzzy existente nos modelos. Além disso, mesmo com um conjunto de ações menos favorável à solução do problema (A1), o processo de aprendizado foi bom. O conjunto A1 é considerado menos favorável porque a probabilidade do agente conseguir otimizar todas as ações que devem ser aprendidas, por cada polipartição de cada célula, é menor. Deste modo, a média de passos necessários para o agente alcançar o objetivo foi maior neste caso, como se verifica na primeira linha da Tabela 1. Quando as chances de se obter na saída o valor do impulso (1 ou -1) aumentam, o agente precisa de um número menor de passos para atingir seu objetivo, conforme se verifica nos resultados dos procedimentos usados em A2 e A3.

O melhor resultado foi obtido com a configuração RL-NFHP$_4$. Esta configuração utiliza o conjunto com maior número de ações (A3) e ações com maior valor absoluto, tornando mais provável que se obtenha na saída o valor integral do impulso - nunca superior a $|1|$. Este conjunto tornou mais flexível para os modelos a escolha da ação.

Comparando-se os testes 2 e 3, verifica-se que o resultado em termos de tamanho da estrutura é maior quando o aprendizado é realizado a partir de pontos aleatórios no espaço de estados. Partindo-se, em cada ciclo, de pontos diferentes do espaço de estados, o crescimento da estrutura é estimulado em pontos diversos da estrutura. Observa-se dos gráficos da Figura 5 que, quando o carro parte do "vale da montanha" (posição -0.5), ele necessita oscilar de um lado para outro para aumentar a sua velocidade, de modo a superar a força da gravidade e atingir o "alto da montanha". Por outro lado, quando o carro parte da posição mais distante (posição -1.2), ele aprende a usar a energia potencial, que neste caso é suficiente para atingir o objetivo sem a necessidade de oscilar.

Verifica-se também que a média do número de passos é maior quando a posição inicial se alterna a cada ciclo entre os pontos -0.5 e -1.2 com velocidade inicial zero. Isto se deve ao fato que, quando a posição inicial é igual a -0.5 e a velocidade é igual a zero, tem-se a condição mais adversa para o aprendizado, elevando a média do número de passos.

A Tabela 2 a seguir compara os resultados obtidos na fase de teste do modelo RL-NFHP com diferentes modelos baseados em RL descritos em Jouffe (1998). A primeira coluna da Ta-

Tabela 1: Configurações e resultados do modelo RL-NFHP aplicados ao carro na montanha

Configurações do Aprendizado					**Resultados Obtidos**		
Modelo	**Pos. inicial**	**Vel. inicial**	**Ações**	**N^o de Ciclos**	**Tamanho da estrutura/ número de parâmetros**	**Média de passos**	
						fase de aprendizado	**fase de teste**
RL-NFHP$_1$	alternada -1.2 -0.5	0	A1={1,0,-1}	3000	121/242	221	101
RL-NFHP$_2$	alternada -1.2 -0.5	0	A2={1,0,-1}*	3000	63/126	151	85
RL-NFHP$_3$	Aleatória [-1.2,0.5]	aleatória [-0.07,0.07]	A2={1,0,-1}*	5000	135/270	100	91
RL-NFHP$_4$	aleatória [-1.2,0.5]	aleatória [-0.07,0.07]	A3={-10,-5,-1,0,1,5,10}**	5000	96/192	91	**69**

*utilizando na saída uma função de discretização

** o impulso aplicado (ação resultante) não é superior a 1 nem inferior a -1 (valores do impulso aplicado ao carro a cada intervalo de tempo).

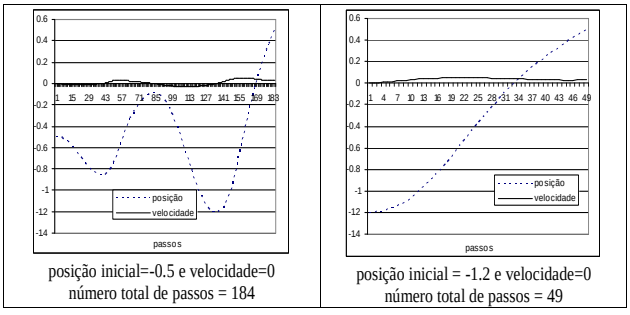


Figura 5: Resultados dos testes para o modelo RL-NFHP4

Tabela 2: Comparação do modelo RL-NFHP com outros modelos (Jouffe, 1998) no caso teste mountain-car problem

Modelo	**no. de parâmetros iniciais**	**Fase de prendizado**	**Fase de Teste**
Neural-Q-learning	4	1724	2189
CMAC *Q-learning*	343	262	85
FQL	25	112	61
RL-NFHP	0 (sem características definidas previamente)	91	69

bela 2 indica o modelo utilizado, a segunda coluna indica o número de parâmetros, ou seja, o número de neurônios na camada escondida para o caso do Neural Q-Learning, o número de grids para o caso CMAC e o número de regras para o FQL. Cada uma destas informações são pré-definidas em seus respectivos modelos. A terceira e quarta coluna indicam o desempenho médio (em 10 experimentos) obtido para cada modelo em termos de números de passos. Estes valores são a média de número de passos em 10 experimentos com o carro partindo de qualquer x $\in$ [-1.2,0.5] e v $\in$ [-0.07,0.07].

A Rede Neural teve o pior desempenho, uma vez que além do aprendizado dos valores Q, ela também precisa aprender a identificar os estados, da mesma forma que o modelo proposto neste artigo; porém as modificações feitas nos pesos da rede pelo algoritmo back propagation afetam toda a rede, prejudicando o aprendizado (Moore, 1991).

A percepção sobre o conjunto de estados ativos pelo modelo CMAC é discreta (muitos estados vizinhos ativam os mesmos conjuntos de grids); já no caso dos modelos FQL e RL-NFHP a percepção sobre o conjunto de estados é contínua. Por esse motivo os resultados dos modelos FQL, RL-NFHP apresentam resultados superiores (Jouffe, 1998).

Nos modelos CMAC e FQL os estados são fixados a priori (grids e regras) e as modificações das funções de valor que ocorrem para o caso CMAC e FQL afetam somente as regras ou os grids que estão ativos em cada passo do aprendizado. Apesar do resultado do FQL ser um pouco melhor que o resultado obtido pelo modelo RL-NFHP, ele parte, assim como o CMAC, de informações prévias (regras e conjuntos fuzzy) relativas ao processo de aprendizado. Em relação ao mo-

delo FQL, embora o modelo RL-NFHP utilize uma função de retorno mais elaborada, ela diz respeito ao objetivo que se deseja alcançar e não ao aprendizado.

A estrutura do modelo RL-NFHP atinge em média 52 células, o que totaliza 104 parâmetros a e b (parâmetros que definem os conjuntos fuzzy associado a cada célula) ao final do processo de aprendizado. Em média, o número de regras ativas a cada estado é 2% do número total de células.

É importante ressaltar que o número de características do modelo RL-NFHP foi considerado igual a zero porque o modelo inicia seu aprendizado com uma única célula e todas, sem exceção, são inicializadas com o mesmo procedimento (para todos os testes realizados), ou seja, os parâmetros a e b são gerados automaticamente, sem nenhuma interferência do usuário. Assim, estes parâmetros não foram considerados como parâmetros de entrada do sistema. Isso vale não apenas para os conjuntos fuzzy iniciais definidos pela primeira célula, mas para todos os outros que serão criados ao longo do processo de aprendizado. Vale lembrar também que a estrutura do modelo (número de células) não é definida previamente.

3.2 Estacionamento do carro (Cart-Centering Problem)

O segundo caso de estudo, o estacionamento do carro (cart-centering problem), é normalmente usado como *benchmarck* da área de programação evolucionária, onde a força aplicada ao carro é do tipo "bang-bang" (Koza,1992). Neste trabalho, a utilização deste estudo de caso teve por objetivo avaliar o aprendizado do modelo RL-NFHP e sua capacidade de controle em diversas condições distintas, além de demonstrar a sua capacidade de se adaptar a mudanças nos limites dos domínios das variáveis de entrada, sem a necessidade de realizar nova fase de aprendizado.

O problema consiste em estacionar, no centro de um ambiente unidimensional, um carro com massa m que se desloca ao longo deste ambiente devido à aplicação de uma força F. As variáveis de entrada para este caso são a posição (x) e a velocidade (v) do carro. O objetivo é estacioná-lo na posição x = 0 com velocidade v = 0. As equações de movimento são:

$$\begin{aligned} x_{t+\tau} &= x_t + \tau.v_t \\ v_{t+\tau} &= v_t + \tau.F_t/m \end{aligned} \qquad (14)$$

onde τ é o tempo ou passo do algoritmo.

A função de retorno do ambiente foi baseada na função de avaliação dada pela equação abaixo:

Se (x > 0 e v < 0) ou (x < 0 e v > 0)

$$R = k_1 e^{(-|distância\_objetivo|)} + k_2 e^{(|velocidade|)} \qquad (15)$$

A função de avaliação da posição e da velocidade cresce à medida que o carro se aproxima do centro do ambiente com velocidade zero. Os valores k_1 e k_2 são constantes maiores do que 1 utilizadas para adequar os valores de retorno à estrutura do modelo.

Os parâmetros γ, $\in$-*greedy* e alfa-cut tiveram seus valores mantidos iguais aos usados nos testes para o carro na montanha. Além disso, os valores utilizados de passo e da massa do carro foram, respectivamente, τ = 0.02 e m = 2.0.

O critério de parada é alcançado entre os valores da velocidade e da posição em relação ao objetivo (x=0 e v=0) forem menores do que 5% do universo de discurso das entradas posição e velocidade. A Tabela 3 mostra a média dos resultados obtidos durante o aprendizado em 5 experimentos, para cada configuração do modelo.

Na Tabela 3, a primeira coluna indica a configuração do modelo testado; os campos limites de posição e velocidade são os limites impostos às variáveis de estado posição e velocidade durante o aprendizado e teste. O campo ações mostra os diferentes conjuntos de forças usadas em cada experimento: F1 = {-150, -75, -50, -30, -20, -10, -5, 0, 5, 10, 20, 30, 50, 75, 150} e F2 = {-150, -75, 0, 75, 150}. Da mesma forma que no caso anterior, a coluna tamanho da estrutura/número de parâmetros exibe a média do número de células ao final de cada experimento e o respectivo número de parâmetros (parâmetros a e b que definem os conjuntos fuzzy). A última coluna mostra a média de passos durante o aprendizado. O número de ciclos foi fixado em 1000. A cada ciclo os pontos de partida do carro eram sempre x=-3 ou x=3.

Tabela 3: Configurações e resultados do modelo RL-NFHP aplicado ao estacionamento do carro

Configurações				Resultados Obtidos	
Modelo	Limites Posição	Limites Velocid.	Ações	Tamanho estrutura/n^o parâmetros	Média de passos fase de aprendizado
RL-NFHP$_1$	|10|	|10|	F1	140/280	221
RL-NFHP$_2$	|3|	|3|	F1	251/502	145
RL-NFHP$_3$	|3|	|3|	F2	193/386	186

A Tabela 4 apresenta a média de passos dos modelos na fase

de teste, partindo dos pontos |3|, |2| e |1| com velocidade zero. É importante ressaltar que estes valores são a média sobre os testes realizados para os 5 experimentos, e que o modelo foi testado em condições iniciais de posição diversas àquelas utilizadas durante o treinamento (|2| e |1|), demonstrando sua capacidade de generalização de conhecimento.

Tabela 4: Resultados da fase de teste do modelo RL-NFHP aplicados ao estacionamento do carro

Modelo	**Média de passos**		
	| 3|	|2|	|1|
RL-NFHP$_1$	190	166	99
RL-NFHP$_2$	109	97	112
RL-NFHP$_3$	96	124	68

Analisando as Tabelas 3 e 4 percebe-se que o modelo RL-NFHP foi capaz de estacionar o carro no ponto central do ambiente, demonstrando o aprendizado do comportamento esperado, nas diversas configurações testadas. A Figura 6 apresenta gráficos de testes realizados com a configuração RL-NFHP$_2$, mostrada na Tabela 3, demonstrando a capacidade do modelo de generalizar seu comportamento convergindo para a posição correta mesmo quando o carro parte dos pontos (-2 e 2) não usados na fase de aprendizado.

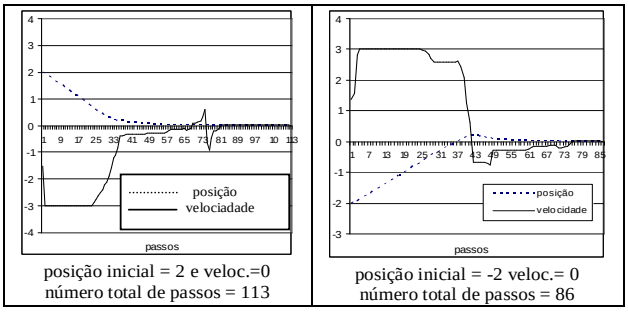
Com o objetivo de avaliar a capacidade do modelo de se adaptar a mudanças nos limites dos domínios das variáveis de entrada, sem a necessidade de realizar novo aprendizado, o modelo gerado com a configuração RL-NFHP2 foi testado com os limites do ambiente ampliados para 10, mantendo-se os limites de velocidade em 3 (Figura 7).

Nestes dois testes, apesar do ambiente ser até mais do que três vezes maior que o ambiente usado durante o aprendizado, o carro conseguiu atingir seu objetivo partindo dos seguintes pontos: no limite do ambiente (10); na metade do ambiente (5). Este resultado foi possível devido à normalização das variáveis de entrada, a qual permite que o aprendizado seja capaz de se adaptar a mudanças no ambiente, sem haver necessidade de novo processo de aprendizado.

3.3 Pêndulo Invertido

O pêndulo invertido pode ser descrito como uma haste (pêndulo) presa a um carro que pode se mover em uma dimensão. O pêndulo move-se em um plano vertical e uma força deve ser aplicada ao carro em intervalos discretos de tempo. Este problema tem duplo objetivo: tentar equilibrar a haste através da força aplicada ao carro, e não se afastar do centro do ambiente.

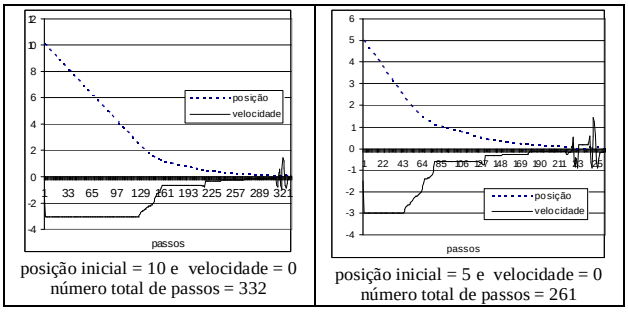
O pêndulo invertido possui 4 variáveis de entrada, fazendo



posição inicial = 2 e veloc.=0
número total de passos = 113

posição inicial = -2 veloc.= 0
número total de passos = 86

Figura 6: Resultados dos testes com o modelo RL-NFHP2



posição inicial = 10 e velocidade = 0
número total de passos = 332

posição inicial = 5 e velocidade = 0
número total de passos = 261

Figura 7: Resultados dos testes com ambiente ampliado para o modelo RL-NFHP2

com que cada nova célula gerada tenha 16 conseqüentes:

θ : ângulo do pêndulo com a vertical;

ω : velocidade angular do pêndulo;

x : posição do carro no ambiente; e

v : velocidade do carro.

A dinâmica do pêndulo invertido foi modelada segundo as equações diferenciais não lineares definidas em (Jouffe, 1998).

Os parâmetros γ e $\in$-*greedy* tiveram seus valores mantidos iguais aos usados nos testes anteriores. O conjunto de ações foi definido como sendo {-10,-5, 0, 5, 10} e o alfa cut=0.0001.

A função de avaliação para o pêndulo invertido é apresentada a seguir:

SE (angulo>=0 e velocidade_carro>=0 e velocidade _angular<=0) ou

SE (angulo<=0) e (velocidade _ carro <=0) e (velocidade _angular>=0)

ENTÃO

$$R = k_1 e^{(-|\ distância\_angular|)} + k_2 e^{(-|posição\_euclidiana|)} \quad (16)$$

SE NÃO R = 0

O valor da função de avaliação cresce à medida que o ângulo e posição se aproximam de zero. Da mesma forma que nos casos anteriores, k_1 e k_2 são constantes maiores do que 1 utilizadas para adequar os valores de retorno à estrutura do modelo.

Ao final do processo de aprendizado, a estrutura do modelo RL-NFHP apresentou 780 células. Este número de células resulta em um total de 1560 parâmetros a e b.

O modelo RL-NFHP foi comparado com o modelo FACL (Jouffe, 1998). Este modelo tem a seguinte configuração: 54 regras, 3 funções de pertinência triangulares para a variável de entrada angular, 3 funções de pertinência trapezoidais para a velocidade angular e para a variável posição, e outras duas funções triangulares para a velocidade. Ambos os modelos foram testados para as condições iniciais de posição angular (θ), velocidade angular (ω), posição euclidiana (x) e de velocidade euclidiana (v) iguais a zero. O modelo FACL executou seus testes de forma semelhante ao executado pelo modelo RL-NFHP, exceto pelo fato de seu sistema ter fixado o número de passos em 500.000. Deste modo, o ciclo só era terminado quando atingia esta meta ou quando o sistema falhava devido ao ângulo ou à posição atingirem valores superiores aos limites definidos acima. O número de ciclos necessários para o aprendizado utilizando-se o modelo FACL foi de 60 enquanto que, para o RL-NFHP, foram necessários 15000 ciclos. Levando-se em conta que o modelo RL-NFHP não tem qualquer conhecimento prévio sobre as regras ou conjuntos fuzzy que modelam os antecedentes das regras, o resultado alcançado pelo modelo RL-NFHP foi satisfatório. Após o aprendizado o pêndulo, verificou-se que o pêndulo oscila dentro do intervalo (-0.006, 0.002) e o carro oscila em torno do ponto central entre os limites (-0.002, 0.004). As velocidades angular (do pêndulo) e do carro também estão dentro dos limites permitidos.

A Figura 8 e a Figura 9 apresentam separadamente a dinâmica do carro e do pêndulo (normalizado para permitir melhor visualização) de um trecho com 350 passos extraídos do teste realizado.

Dessa forma, verifica-se através dos gráficos anteriores que o modelo foi capaz de controlar o pêndulo invertido sem deixá-lo cair.

3.4 Aplicação em robótica: KHEPERA2

A última aplicação baseia-se no simulador do robô Khepera, o qual possui seis variáveis de entrada. O simulador foi implementado com base no simulador desenvolvido por Ostergard (2000) para o robô Khepera.

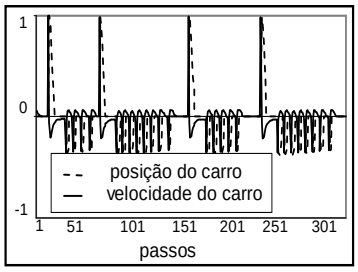


Figura 8: Dinâmica do carro com o modelo RL-NFHP

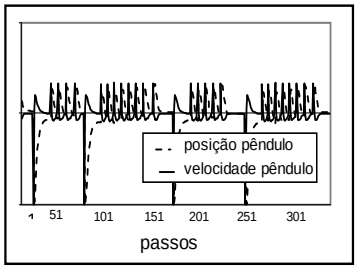


Figura 9: Dinâmica do pêndulo com o modelo RL-NFHP

O robô Khepera adquire sinais do ambiente a paratir de 8 sensores grupados dois a dois: um à frente, um atrás, um à esquerda e um à direita. Assim como o robô Khepera, este simulador possui 2 motores independentes, cujas potências variam entre -20 e 20 (Dias, 2004).

Outras informações importantes para o robô são a distância euclidiana entre sua posição atual e seu objetivo e a distância angular entre a frente do robô e o objetivo.

O movimento do robô é definido no espaço pela posição (x,y) (orientação euclidiana) e pelo ângulo α (orientação angular). A rotação (equação (17)) e a translação (equação (18)) do robô são calculadas com base nas velocidades dos motores direito e esquerdo (Ostergard, 2000). Os números apresentados no denominador das equações (17) e (18) têm por objetivo adequar os valores de rotação e de translação.

$$\text{rotação} = \frac{(vel\_esq - vel\_dir)}{12}\ (graus) \quad (17)$$

$$\text{translação} = \frac{(vel\_esq + vel\_dir)}{20}\ (unid.\ de\ velocidade) \quad (18)$$

Os valores resultantes da rotação e da translação, a cada passo, definem as novas posições angular (equação (19)) e euclidiana (equação (20)) (Ostergard, 2000).

$$\alpha_{t+1} = \alpha_t + rotação \quad (19)$$

$$x_{t+1} = x_t + translação.sen(\alpha_{t+1})$$
$$y_{t+1} = y_t + translação.\cos(\alpha_{t+1}) \quad (20)$$

As seis variáveis de entrada que o robô Khepera possui são, portanto:

- O ângulo (β) entre o robô e o objetivo em radianos;
- A distância ao objetivo calculada a partir das coordenadas dadas pelo simulador; e
- A leitura dos 4 sensores distribuídos ao redor do robô (direita, esquerda, à frente, atrás), os quais podem ter um valor entre 0 (nenhum obstáculo) e 1024 (tocando o obstáculo).

Como este problema tem 6 entradas, cada nova célula terá 64 conseqüentes.

A saída do sistema são as ações que cada agente deve executar para atingir seus objetivos. Para o caso do robô Khepera, são as potências aplicadas, a cada passo, nos motores (esquerdo e direito), tendo seus valores entre -20 (velocidade máxima para trás) e 20 (velocidade máxima para frente). O conjunto de ações contendo as potências combinadas dos dois motores (direito e esquerdo) foi: [(-20,-20), (-10,-10), (-5,-5), (0,0), (20,20), (10,10), (5,5), (-20,20), (-10,10), (-5,5), (20,-20), (10,-10), (5,-5)]. Os três primeiros elementos deste conjunto resultam em movimento para trás; o quarto, em motores parados; o quinto, sexto e sétimo resultam em movimento para frente; o oitavo, nono e décimo, rotação para a esquerda, e os demais elementos, rotação para a direita.

A função de reforço (equação (21)) do ambiente é composta por uma parcela relativa à distância angular, outra relativa à distância euclidiana e uma última que corresponde à proximidade do obstáculo. As funções relativas à distância euclidiana e angular são do tipo exponenciais. Na equação (21) são também utilizadas constantes (k_1 e k_2) maiores que 1 para adequar os valores de retorno à estrutura do modelo.

$$R = (k_1 e^{-(dist\_euclidiana)} + k_2 e^{-(|dist\_angular|)}) * (1 - distância\_obstáculo/1024) \quad (21)$$

A componente de avaliação angular abrange os ângulos dos intervalos (-180˚, 0˚) e (0˚, 180˚); quanto menor o ângulo β (distância angular), maior será o valor do reforço. Esta componente tem peso um pouco maior que a avaliação relativa à distância euclidiana, de forma a incentivar que o robô use preferencialmente o movimento frontal. A avaliação relativa à distância euclidiana também aumenta seu valor em função da proximidade ao objetivo. Já a componente de retorno associada à proximidade do obstáculo é calculada em termos percentuais, em uma escala que vai de 0 a 1024 (Ostergard, 2000). Esta função só é considerada quando os sensores atingem valores superiores a 600 para o sensor frontal e 900 para os demais. Quanto mais próximo do obstáculo, maior o percentual representado por esta função. O objetivo desta avaliação relativa ao obstáculo é reduzir globalmente o valor de retorno quando o robô se aproxima do obstáculo.

Os parâmetros γ e $\in$-*greedy* tiveram seus valores mantidos iguais aos usados nos testes anteriores, e alfa- cut=0.0001.

Para este estudo de caso foram realizados dois tipos de experimentos: o primeiro sem obstáculo (Figura 10(a)) e o segundo considerando um obstáculo que é apresentado no centro do ambiente na Figura 10(b). Ao final do processo de aprendizado nos dois tipos de ambiente, as estruturas geradas atingiram 498 e 812 células e o número de ciclos usados foi em média de 5000 e 10000, respectivamente, para o caso sem e com obstáculo. Este número de células resulta em um total de 996 e 1624 parâmetros a e b para o ambiente sem e com obstáculos, respectivamente.

A Figura 10 mostra os resultados obtidos com o robô Khepera na fase de teste, partindo de diferentes ângulos e posições deste ambiente. Os círculos brancos mostram os pontos de partida do robô Khepera e os círculos cinzas mostram os pontos definidos como objetivo; o erro permitido foi de 5% em relação à maior distância euclidiana do ambiente.

Os pontos de partida usados durante o aprendizado foram (-2, -2) com ângulos α alternando a cada ciclo: α = 90˚ e α = 0˚. No entanto, nota-se que mesmo partindo de pontos diferentes dos usados durante a fase de aprendizado, o robô consegue atingir seu objetivo. Estes testes também envolveram a ângulos diferentes. Para os testes sem obstáculos, as seguintes configurações foram usadas: o robô 1 partiu com ângulo α=45˚; o 2 partiu com ângulo α=45˚; o 3 partiu com ângulo α=90˚; o 4 partiu com ângulo α=90˚; o 5 partiu com ângulo α=-135˚ e o número 6 partiu de α=-90˚. O robô número 5 foi o que apresentou o pior desempenho; primeiro moveu-se para frente (a sua frente estava voltada para o canto) para em seguida, fazer uma rotação para o lado direito e, finalmente, dirigir-se para o objetivo.

Para o teste com obstáculos, o robô marcado com o número 1 partiu com ângulo α = 45˚; o número 2 partiu com ângulo α=45˚; o 3 partiu com ângulo α=-90˚; o 4 partiu com ângulo α=90˚; o 5 com ângulo α=90˚; o 6 com ângulo α=0˚;

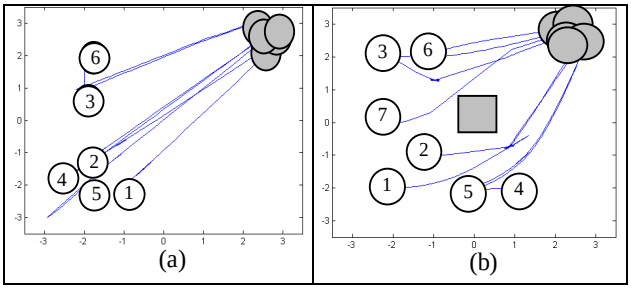


Figura 10: Resultados do robô (a) sem e (b) com obstáculos

e, finalmente, o robô número 7 partiu com ângulo α=0˚. Os ângulos definidos para os robôs 1, 2, 5 e 7 tiveram por objetivo verificar o comportamento do robô nas condições mais adversas (o obstáculo no meio do caminho). O robô 4 apresentou o pior desempenho, uma vez que, partindo da posição (1, -2) com ângulo α = 90˚, caminhou primeiramente para o ponto (0,-2), para depois se dirigir ao objetivo. O comportamento demonstrado, de uma forma geral, foi o de girar sobre o próprio eixo antes de caminhar na direção do objetivo.

Pode-se concluir que o modelo RL-NFHP também demonstrou ser capaz de aprender o comportamento desejado, mesmo quando submetido a um problema com um número maior de variáveis de entradas, o que torna ainda mais complexa a capacidade de aprendizado devido ao alto número de conseqüentes.

Como esperado, no caso sem obstáculos obtém-se uma estrutura menor do que naquele com obstáculos. Isto se deve à maior complexidade nas posições próximas ao obstáculo. Pelo mesmo motivo, o número de ciclos necessários para o aprendizado, no caso sem obstáculos, foi também menor.

4 CONCLUSÕES

Este artigo apresentou em detalhes a proposta de um novo modelo hierárquico neuro-fuzzy baseado no aprendizado por reforço, denominado RL-NFHP. O modelo RL-NFHP é capaz de criar e expandir a estrutura de regras sem qualquer conhecimento a priori (regras ou conjuntos fuzzy); extrair conhecimento a partir da interação direta do agente com ambientes grandes e/ou contínuos, utilizando aprendizado por reforço, de modo a aprender quais ações devem ser executadas; e produzir resultados lingüisticamente interpretáveis, sob a forma de regras fuzzy, que constituam o raciocínio que o agente deve inferir para atingir sua(s) meta(s).

O modelo RL-NFHP foi avaliado em diferentes estudos de casos e demonstrou ser capaz de generalizar suas ações, mostrando comportamento adequado quando o agente se encontra em estados cujas ações não foram especificamente aprendidas. Esta capacidade amplia a autonomia do agente. A normalização das entradas também permitiu que o modelo respondesse adequadamente a mudanças nos limites das variáveis de entrada. Esta característica também foi bastante interessante, pois aumenta ainda mais a autonomia desejada para este agente.

Um ponto que merece ser destacado é que o modelo RL-NFHP foi capaz de realizar o aprendizado interagindo diretamente com o ambiente, sem que houvesse qualquer processamento off-line e sem usar qualquer informação que pudesse beneficiar o aprendizado. No entanto, o modelo pode ser capaz de aceitar, caso necessário, a inclusão de informações prévias.

É importante ressaltar que o RL-NFHP dispensou a análise prévia das variáveis de entrada, não requerendo ordenação nem priorização destas variáveis. Isto porque as células de sua estrutura foram elaboradas com a capacidade de suportar todas as entradas simultaneamente. Isto é bastante importante pois, na maioria dos problemas cuja solução é baseada em RL, é inviável avaliar o comportamento do agente através da utilização individual das entradas. Além disso, a análise prévia das entradas aumenta o tempo computacional para o aprendizado do agente.

A principal desvantagem do modelo proposto é que a função de retorno precisa ser mais elaborada do que as funções requeridas por outros modelos NF baseados em RL pois, sem esta informação, a solução seria extremamente difícil de ser encontrada, uma vez que o sistema precisa, paralelamente ao aprendizado do comportamento do agente (ações), aprender a identificar seus estados (estrutura de regras).

Este estudo de casos confirma a boa aplicabilidade do modelo RL-NFHP na área de controle e robótica, encorajando o prosseguimento da pesquisa de aprendizado automático utilizando Sistemas Neuro-Fuzzy Hierárquicos.

Uma proposta de aperfeiçoamento do modelo RL-NFHP é a utilização do princípio WoLF (Win or Learn Fast) (Bowling e Veloso, 2001) para modificar a taxa de aprendizado que ajusta a política. O princípio WoLF consiste em aprender rapidamente quando se está perdendo e mais devagar quando se está ganhando.

O modelo também está sendo modificado para ser utilizado em ambientes multiagentes cooperativos (Kapetanakis, 2002). Nestes ambientes o aprendizado dos agentes ocorre não somente através de tentativa e erro, mas também através do compartilhamento de informações instantâneas, episódios e conhecimento adquirido. No caso dos RL-NFHP, a estrutura (formada pelo conjunto de células) poderá ser compartilhada entre os agentes. Dessa forma, a adaptação do RL-NFHP para sistemas multiagentes cooperativos permitirá

que agentes troquem informações a respeito de sua estrutura, reduzindo o tempo para o aprendizado.

REFERÊNCIAS

Barto, A. G. and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):41–77.

Arvo, J. (1988). Linear Time Voxel Walking for Octrees. *Ray Tracing News*.

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

Bowling, M. and Veloso, M. (2001). Rational and convergent learning in stochastic games, In Proceedings of the Seventeenth International Joint Conf. on Artificial Intelligence, pp. 1021-1026, Seattle, US.

Brenner, W., Zarnekow, R. and Wittig, H., (1998). *Intelligent Software Agents*, Springer, 1998.

Dias, P., Figueiredo, K., Vellasco, M.M.B.R., Pacheco, M.A.C. and Barbosa, C.R.H. (2004), Reinforcement Learning Hierarchical Neuro-Fuzzy Model for Autonomous Robots, Proc. of the 2nd Inter. Conf. on Autonomous Robots and Agents, N.Z., Dec. 13-15.

Figueiredo, K., Vellasco, M., 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),pp.130-135, (ISBN 0-7695-2291-2), IEEE Comp. Society, Japan, Dec.,2004.

Figueiredo, K., Vellasco, M. and Pacheco, M. A. (2005a). Hierarchical Neuro-Fuzzy Models basedon Reinforcement Learning for Intelligent Agents. In Proceedings 8th International Work-Conference on Artificial Neural Networks (IWANN'2005), Spain.

Figueiredo, K, Vellasco, M., Pacheco, M.A. and Souza, F.J. (2005b). Modified Reinforcement Learning-Hierarchical Neuro-Fuzzy Politree Model for Control of Autonomous Agents. International Journal of Simulation Sys., Sci.&Tech, UK, v.6, n. 10-11, p. 4-13.

Gonçalves, L.B., Vellasco, M., Pacheco, M.A.C. and Souza, F.J. (2006). Inverted Hierarchical Neuro-Fuzzy BSP System: A Novel Neuro-Fuzzy Model for Pattern Classification and Rule Extraction in Databases, aceito para publicação no IEEE Trans. on Sys., Man & Cyber., Part C: Applications and Review, (ISSN: 1094-6977), IEEE, Vol. 36, No. 2, pp. 236-248, March.

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

Jang, J.S.R. (1993). ANFIS: Adaptive-Netwo rk-Based Fuzzy Inference System, *IEEE Trans. on Sys., Man, and Cybernetics*, Vol.23, N.3, may/june 1993, p.665-685.

Jouffe, L. (1998). Fuzzy Inference System Learning by Reinforcement Methods, *IEEE Trans. on Sys., Man and Cybernetics*, part c, vol.28, n. 3, pp. 338-355.

Kapetanakis, S. and Kudenko, D. (2002). Reinforcement Learning of Coordination in Cooperative Multi-agent Systems. In Proceedings of AAAI-02/IAAI-02. Edmonton, Alberta.

Koza, J.R. (1992). Genetic Programming: On the programming of computers by means of natural selection, Cambridge, MA, MIT Press.

Kruse, R.and Nauck, D. (1995). NEFCLASS-A Neuro-Fuzzy Approach for the Classification of Data, Proc. of the ACM Symposium on Applied Computing, Nashville.

Lima, M.J.L., Melo, J.D. and Doria Neto, A.D., (2005). Hierarchical Reinforcement Learning for Metrical Task Systems, Fifth International Conference on Hybrid Intelligent Systems (HIS'05), pp. 251-258, RJ, Brazil.

Mendel, J.M. (1995). Fuzzy Logic Systems for Engineering: A Tutorial, Proceedings of the IEEE, Vol.83 , n. 3, march, p. 345-377.

Moore, A.W. (1991). Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces, Proc. of the Eighth Inter. Conf. on Machine Learning, L.A., Morgan Kaufmann, 333-337.

Ostergard, E.H. (2000). Evolving Complex Robot Behaviour, Master's Thesis of Departmente of Computer Science, University of Aarhus, Denmark, May.

Ribeiro, C.H.C. (1999) A Tutorial on Reinforcement Learning Techniques In: Inter. Joint Conf. on Neural Networks ed: INNS Press.

Souza, F.J., Vellasco, M.M.B.R., Pacheco, M.A. (2000). Hierarchical Neuro-Fuzzy BSP Model – HNFB, Proceedings of the VIth Brazilian Symp. on Neural Networks – Volume 2 – (ISBN 0-7695-0856-1), IEEE Computer Society, Rio de Janeiro, RJ, 22-25 Nov.

Souza, F.J., Vellasco, M. and Pacheco, M.A. (2002). Hierarchical Neuro-Fuzzy QuadTree Models, FUZZY SETS & SYSTEMS, ISSN 0165-0114, vol 130/2, August 2002, Elsevier Science, The Netherlands, pp. 189-205.

Sutton, R.S.(1996). Generalization in Reinforcement learning: Succesful examples using sparse coarse coding, In Touretzky, D., Mozer, M., and Hasselmo, M., ed., Advances in Neural Information Proc. Sys.8, pp.1038-1044, MIT Press.

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

Vuorimaa, P. (1994). Fuzzy self-organing map, *Fuzzy Sets and Systems*, No.66, pp.223-231.