APRENDIZADO POR REFORÇO NO DOMÍNIO DO FUTEBOL DE ROBÔS 2D: UMA ANÁLISE DO TRAÇO DE ELEGIBILIDADE COM UMA COMPARAÇÃO ENTRE OS ALGORITMOS Q-LEARNING(λ) E SARSA(λ).

Rubisson Duarte Lamperti*, Erivelton Geraldo Nepomuceno*, André Luiz C. Ottoni*

* *UFSJ - Universidade Federal de São João del-Rei*
Campus Santo Antônio, Praça Frei Orlando, 170
São João del-Rei, MG, Brasil

Emails: duartelamperti@yahoo.com.br, nepomuceno@ufsj.edu.br, andreottoni@ymail.com

Abstract— The Reinforcement Learning (RL) is a technique in which the agent learns through direct interaction with the environment. One advantage of reinforcement learning is the ability of the agent to learn in an unknown environment. However, the lack of knowledge of the environment, the complexity of the system and a large state space contribute to increasing time. In this study we used the Eligibility Trace in an attempt to reduce learning time in the field of robot soccer 2d. Q-Learning (λ) and Sarsa(λ) were used to evaluate the reinforcement learning in set with the trace for the training of attack against defense in the simulation. It was found that the Eligibility Trace affect the behavior of both algorithms, and Q-Learning(λ) performed better than Sarsa(λ).

Keywords— Reinforcement Learning in Control Systems, Multiagent System, Robocup.

Resumo— O Aprendizado por Reforço (AR) é uma técnica em que o agente aprende por meio da interação direta com o ambiente. Uma das vantagens do aprendizado por reforço é a possibilidade do agente aprender em um ambiente desconhecido. Porém, a falta de conhecimento do ambiente, a complexidade do sistema e um grande espaço de estados contribuiem para o aumento do tempo de aprendizado. Neste trabalho foi utilizado o Traço de Elegibilidade na tentativa de reduzir o tempo de aprendizado no domínio do futebol de robôs 2d. Os algoritmos Q-Learning(λ) e Sarsa(λ) foram utilizados para avaliar o aprendizado por reforço em conjunto com o Traço para o treinamento de ataque contra defesa no ambiente de simulação 2d. Verificou-se que o Traço de Elegibilidade influenciou no comportamento de ambos os algoritmos, e que Q-Learning(λ) teve melhor desempenho que Sarsa(λ).

Keywords— Aprendizagem por Reforço em Sistemas de Controle, Sistema Multiagente, Robocup.

1 Introdução

A Inteligência Artificial (IA) é uma área da ciência que abrange conceitos como, autonomia e inteligência de máquinas. Dentro da IA existem várias técnicas, tais como, o Aprendizado por Reforço (AR) (Silva et al., 2010), o Aprendizado por Reforço Acelerado por Heurísticas (Junior, 2007), e as Redes Neurais Artificiais (Simões et al., 2007).

O AR é uma técnica em que o agente aprende por meio de interação direta com o ambiente (Sutton, 1988). Uma das vantagens do uso da técnica de AR está na possibilidade do agente aprender em um ambiente totalmente desconhecido. Entretanto, a falta de conhecimento prévio do ambiente e um grande número de variáveis de estados podem contribuir para o aumento no tempo de aprendizado (Sherstov and Stone, 2005).

A técnica denominada por Traço de Elegibilidade (Klopf, 1972) em conjunto com AR tem sido amplamente utilizada na tentativa de diminuir o tempo de aprendizado por meio da memorização dos pares estado/ação recém-visitados.

A simulação de futebol de robôs 2D é uma categoria criada pela Robocup[1], na qual, partidas de futebol entre equipes de robôs virtuais autônomos são realizadas para simular condições reais de jogo.

[1]Robocup Federation: http://www.robocup.org

A complexidade do ambiente 2D é uma das características que torna o domínio de futebol de robôs adequado para a avaliação das técnicas de AR em conjunto com o Traço de Elegibilidade.

Este trabalho apresenta a modelagem de uma estratégia de aprendizado por reforço para o domínio multiagente do futebol de robôs 2D da RoboCup. Além disso, são comparados os algoritmos Q-learning (Watkins, 1989) e Sarsa (Sutton and Barto, 1998) em conjunto com o traço de elegibilidade.

Este artigo está organizado como segue: Seção 2 - introdução de alguns conceitos de AR, tais como, Q-learning e Sarsa, Traço de Elegibilidade, e uma breve introdução sobre a plataforma de futebol de robôs simulado 2D; Seção 3 - implementação do modelo proposto para o aprendizado; Seção 4 - análise dos resultados; e, finalmente, a conclusão final.

2 Fundamentação Teórica

2.1 Algoritmo Q-Learning

O algoritmo Q-Learning (Watkins, 1989) é conhecido por vários autores como sendo de fácil implementação (Kaelbling et al., 1996), e por isso, tem sido amplamente utilizado (F. et al., 2008).

O Q-Learning tem como principal caracterís-

tica a capacidade de aprender por meio da interação de uma política π^* ótima na ausência de um modelo do sistema. A ideia do algoritmo é aprender por meio da interação com o ambiente, utilizando as melhores ações para encontrar a função valor ótima $Q^*(s, a)$ (Watkins and Dayan, 1992).

O conhecimento adquirido é armazenado em uma matriz $Q(s, a)$, que possui os valores estimados para cada par de estado-ação. A função Q é estimada por meio da equação (Watkins, 1989):

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

onde,
s é o estado atual; a é a ação executada em a;
r é a recompensa dada ao executar a estando em s; s' é o próximo estado; a' é a ação executada em s'; α é a taxa de aprendizagem;
γ é a taxa de desconto $(0 < \gamma < 1)$.

O termo $\max_{a'} Q(s', a')$ representa a utilização das melhores ações na busca da função valor ótima $Q^*(s, a)$, pois, se utiliza das ações que fornece o melhor valor da função Q. O algoritmo procedural pode ser visto em detalhes abaixo:

Algoritmo: Q-Learning (Off-Policy)
Inicialize $Q(s, a)$ arbitrariamente ;
para todo *episodio* **faça**
  Inicialize s (estado inicial) ;
  repita
    $a \leftarrow$ ação para s utilizando política derivada de Q ;
    Tome ação a, observe a recompensa r e o próximo estado s' ;
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$;
    $s \leftarrow s'$;
  até *s ser terminal*;
fim

2.2 Algoritmo Sarsa

O algoritmo Sarsa utiliza o quíntuplo (s_t, a_t, r_t, s_{t+1}, a_{t+1}), o qual advém o nome *Sarsa* (Sutton and Barto, 1998). A regra de atualização baseada em Sarsa é semelhante à regra de atualização de Q-learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \quad (2)$$

Diferente do algoritmo Q-Learning que aprende os valores da política ótima, Sarsa aprende os valores da política atual, sendo denominado On-Policy, o que significa que a maior recompensa esperada não é necessariamente selecionada para a atualização da função valor estado-ação $Q(s, a)$. O algoritmo procedural para o Sarsa pode ser visto a seguir:

Algoritmo: Sarsa (On-Policy)
Inicialize $Q(s, a)$ arbitrariamente ;
para todo *episodio* **faça**
  Inicialize s (estado inicial) ;
  $a \leftarrow$ ação para s utilizando política derivada de Q ;
  repita
    Tome ação a, observe a recompensa r e o próximo estado s' ;
    $a' \leftarrow$ ação para s' utilizando política derivada de Q ;
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$;
    $s \leftarrow s'$; $a \leftarrow a'$;
  até *s ser terminal*;
fim

2.3 Traço de Elegibilidade

Introduzido por Klopf (Klopf, 1972), a ideia do Traço de Elegibilidade é memorizar em um curto tempo os estados visitados, ou seja, um traço, o qual, decai gradualmente ao longo do tempo, marcando o estado como elegível para o aprendizado.

Geralmente, o Traço de Elegibilidade decaem exponencialmente com o produto entre o parâmetro de decaimento λ, e a taxa de desconto γ. O acúmulo de traços é definido por,

$$e_{t+1}(s) = \begin{cases} \gamma\lambda e_t(s), & s \neq s_t; \\ \gamma\lambda e_t(s) + 1, & s = s_t. \end{cases} \quad (3)$$

onde, e_t representa o Traço de Elegibilidade para o estado s no tempo t, e s_t é o estado atual no tempo t.

Nos problemas de controle, cada par estado-ação tem um traço distinto. Quando um estado é visitado e uma ação é tomada, apenas no traço do estado para aquela ação é acrescido de +1, enquanto os demais traços decaem exponencialmente até zero.

2.4 Futebol de Robôs Simulado 2D

A categoria de simulação 2D da RoboCup foi desenvolvida pelo Dr. Itsuki Noda em 1993 (Boer and Kok, 2002). A simulação 2D é constituída por 11 jogadores virtuais criados por meio de softwares. A simulação de uma partida é feita em um servidor denominado *Soccer Server* (Noda, 1995). Este servidor foi desenvolvido para ser um ambiente de simulação multiagentes em tempo real, sendo este um sistema que simula a complexidade de uma partida de futebol com a adição de ruído (Noda and Stone, 2002).

Trainer

Na RoboCup 2D, dois tipos de treinadores podem ser aplicados ao time: off-line e on-line. O modo off-line, geralmente, é usado para o treinamento, pois, o treinador tem um controle quase total dos jogadores e do jogo, podendo mudar a posição e a velocidade da bola e dos jogadores. No modo on-line, o treinador pode apenas comunicar-se com os jogadores em campo. E em ambos os modos, o treinador pode receber as informações globais sem ruído e usá-las durante o jogo.

3 Implementação do Modelo para o Aprendizado

O ambiente de simulação de futebol de robôs 2D da Robocup foi utilizado na realização dos experimentos. Estes experimentos foram realizados no modo Trainer, possibilitando a simulação da condição de ataque contra defesa entre dois times, sendo o primeiro definido pela cor azul constituído por dois atacantes, e o segundo definido pela cor vermelha e formado por um goleiro e dois zagueiros (Figura 2).

Neste trabalho, os agentes receberam 360 graus de visão para garantir a eles o conhecimento completo do mundo (Stone et al., 2005). Além disso, optou-se em limitar a posição dos jogadores entre os intervalos $25 \leq X \leq 52$ e $-32 \leq Y \leq 32$, conforme ilustrado na Figura 1. Para cada simulação foram realizadas 140 episódios, sendo cada episódio composto por 12 tentativas a gols, totalizando 1680 tentativas a gols. A determinação do número de tentativas a gols por episódio foi feita com base na metodologia proposta por Takahashi (Takahashi, 2007). O início de cada tentativa ocorreu com o sorteio aleatório dos jogadores e o fim foi definido pelas seguintes situações: a posse de bola por um dos jogadores do time defensor (cor vermelha); após chute a gol; e, a saída da bola de dentro da área limitada.

3.1 Estados Discretos

As informações escolhidas para compor o espaço de estados foram a posição do agente, e a distância D entre o agente com a posse de bola e o adversário mais próximo (ver Figura 2). A posição do agente foi definida utilizando as subáreas, AA_1, AA_2, ..., AA_{12}, como mostra a Figura 1. A variável D pertence ao intervalo $[0, 20m[$, a qual, foi discretizada por meio das variáveis d_1 e d_2, conforme pode ser visto no sistema 4.

$$D = \begin{cases} 0 \leq & d_1 & < 10, \\ 10 \leq & d_2 & < 20. \end{cases} \quad (4)$$

Os estados s_i foram formados por uma combinação entre as subáreas, AA_i $(i = 1, 2, ..., 12)$ e as variáveis d_j $(j = 1, 2)$, por exemplo, $s_1 = AA_1 \cap d_1$.

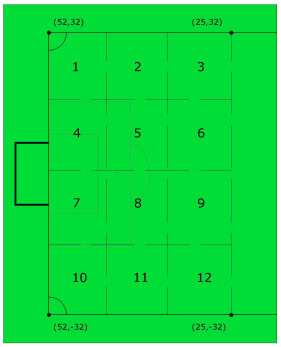

Figura 1: Representação das variáveis de estado AA_i $(i = 1, 2, ..., 12)$.

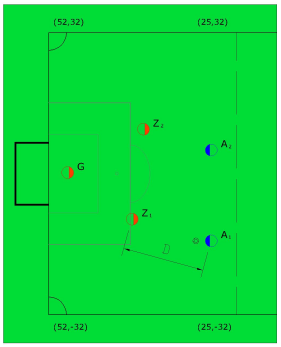

Figura 2: Representação da variável D.

Neste trabalho, o modelo proposto, por meio de estados discretos, foi formado por vinte e quatro $(12x2 = 24)$ estados e quatro ações, totalizando 96 pares estado/ação.

3.2 Ações Discretas

As tomadas de decisões devem ser feitas por meio de ações pré-determinadas, as quais, fazem parte de um conjunto A de ações. Por definição, a ação a_i deverá pertencer ao conjunto A de ações para que a_i seja tomada. No modelo de aprendizagem proposto, as ações contidas em A foram definidas como:

1. a_1 - Segurar a bola (ficar parado com a bola);
2. a_2 - Chute a gol (chutar a bola em direção ao gol);
3. a_3 - Passe (passar a bola para o outro jogador);
4. a_4 - Carregar a bola (deslocar com a bola).

3.3 Recompensas

A decisão tomada pelo agente ao escolher a ação $a \in A$, no estado $s \in S$ resulta no ganho de uma recompensa r, a qual, é dada em função do par estado-ação atual (s_t, a_t) do agente.

Tabela 1: Sarsa(λ) - Tempo de decaimento (episódio).

λ	**Tempo de decaimento (episódio)**
$0,0$	116
$0,1$	118
$0,2$	92
$0,3$	101
$0,4$	106
$0,5$	113
$0,6$	104
$0,7$	103

A função de recompensa foi utilizada na forma tabular, a qual, é denominada de matriz de recompensa R. Os elementos da matriz R, $r_t(s,a)$, informam o valor da recompensa para cada par estado-ação. A matriz R, 24×4, teve seus elementos todos iguais a -1, exceto, nos casos em que a ação terminou em gol, $r_t(s,a) = 50$, ou, com a perda da bola, $r_t(s,a) = -10$.

3.4 Função Valor - $Q(s,a)$

A função valor estado-ação Q foi estimada segundo os algoritmos Sarsa(λ)(2.2) e Q-Learning(λ)(2.1). A função Q foi representada na forma tabular, onde cada elemento da matriz, $Q(s,a)$, representa o valor da função para cada par estado-ação, totalizando 96 elementos.

4 Análise dos Resultados

Inicialmente, os pares estado/ação foram observados na execução do algoritmo Sarsa(λ) e Q-learning(λ), onde verificou-se a visita a todos os pares. A ocorrência das visitas aos estados segue uma das premissas para o aprendizado, a qual, determina que todos os pares estado/ação devem ser visitados infinitas vezes (Watkins, 1989).

4.1 Simulação do Algoritmo Sarsa(λ)

Após a determinação do número de tentativas por episódio, foram realizadas as simulações, nas quais, foram obtidos os resultados da média da diferença temporal $(Q_{t-1} - Q_t)$ da matriz $Q(s,a)$ para os diferentes valores do parâmetro λ (Figura 3). Como ilustrado, o comportamento das curvas foi semelhante para os parâmetros $\lambda's$, diferindo apenas no tempo de decaimento da diferença $Q_{t-1} - Q_t$. As curvas para $\lambda = 0,1$ e $\lambda = 0,2$ apresentaram o maior e o menor tempo, respectivamente, para o decaimento da curva até o ponto de equilíbrio. O número de episódios para que a diferença temporal estabilizasse em torno do ponto de equilíbrio foi igual a 118 e 92 para $\lambda = 0,1$ e $\lambda = 0,2$, respectivamente (Tabela 1).

Ao analisar a média de gols acumulados para os diversos valores de λ verificou-se que todas as

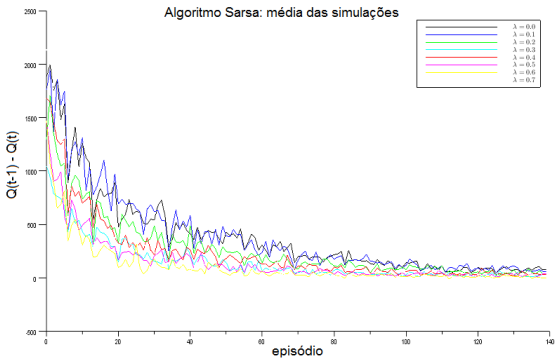

Figura 3: Sarsa(λ) - Média da diferença temporal da matriz $Q(s,a)$ para os $\lambda's$ simulados.

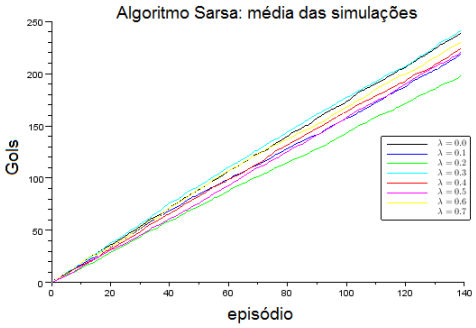

Figura 4: Sarsa(λ) - Média dos gols acumulados para os $\lambda's$ simulados.

curvas foram crescentes durante os 140 episódios (Figura 4), apesar de apresentarem saldos de gols diferentes (Tabela 2), sendo que o maior (241,60 gols) foi para $\lambda = 0,3$ e o menor (198,5 gols) para $\lambda = 0,2$.

4.2 Simulação do Algoritmo Q-Learning(λ)

Neste experimento, os resultados obtidos podem ser vistos por meio da média da diferença temporal $(Q_{t-1} - Q_t)$ da matriz $Q(s,a)$ e a média do número de gols (acumulados e por episódios) para os diferentes valores do parâmetro λ. Para o algoritmo Q-Learning(λ), as curvas que apresentaram o maior e o menor tempo para o decaimento até o ponto de equilíbrio foram para $\lambda = 0,1$

Tabela 2: Sarsa(λ) - Média de Gols por Episódio.

λ	**Gols Acumulados**
$0,0$	239,30 ± 22,69
$0,1$	219,10 ± 12,70
$0,2$	198,50 ± 23,30
$0,3$	241,60 ± 18,12
$0,4$	224,50 ± 20,17
$0,5$	220,80 ± 17,92
$0,6$	230,00 ± 23,49
$0,7$	226,30 ± 23,86

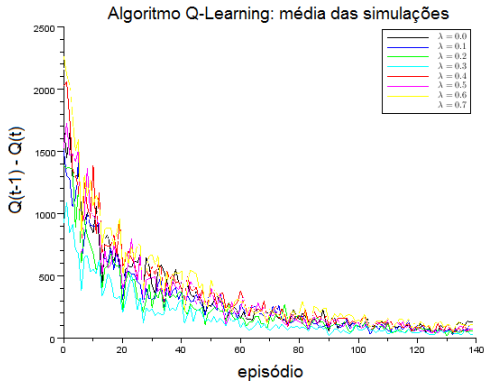

Figura 5: Q-Learning(λ) - Média da diferença temporal da matriz $Q(s,a)$ para os $\lambda's$ simulados.

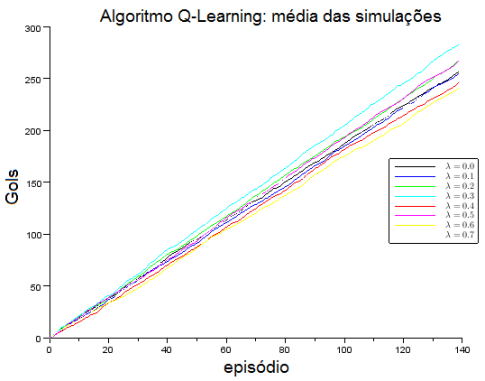

Figura 6: Q-Learning(λ) - Média dos gols acumulados para os $\lambda's$ simulados.

Tabela 3: Q-Learning(λ) - Tempo de decaimento (episódio).

λ	**Tempo de decaimento (episódio)**
$0,0$	105
$0,1$	126
$0,2$	92
$0,3$	88
$0,4$	106
$0,5$	84
$0,6$	45
$0,7$	97

e $\lambda = 0,3$, respectivamente. Para estes valores foram necessários 126 ($\lambda = 0,1$) e 75 ($\lambda = 0,6$) episódios para que a diferença temporal estabilizasse em torno do ponto de equilíbrio.

Do mesmo modo que na simulação de Sarsa(λ), as curvas que representam a média de gols acumulados apresentaram um comportamento crescente (Figura 6) com variações no saldo de gols. O maior e menor número de gols acumulados foi obtido para $\lambda = 0,3$ $(283,20)$ e $\lambda = 0,6$ $(241,40)$, respectivamente (Tabela 3).

4.3 Comparação entre Sarsa(λ) e Q-Learning(λ)

Ao comparar o algoritmo Sarsa(λ) com o Q-Learning(λ) foi possível observar, de acordo com o teste t-student ($p = 1\%$ e $p = 5\%$), que o último apresentou maior número de gols durante todos os episódios (Figura 7 e 8). Em relação ao total de gols acumulados, o algoritmo Q-Learning(λ) apresentou uma superioridade de aproximadamente de 8%.

Tabela 4: Q-Learning(λ) - Média de Gols por Episódio.

λ	**Gols Acumulados**
$0,0$	257,60 ± 8,00
$0,1$	254,90 ± 6,12
$0,2$	267,30 ± 8,67
$0,3$	283,20 ± 6,70
$0,4$	246,50 ± 7,38
$0,5$	267,30 ± 7,12
$0,6$	241,40 ± 6,20
$0,7$	254,80 ± 5,33

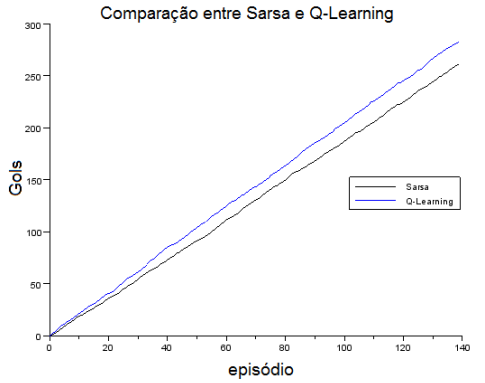

Figura 7: Comparação entre Sarsa(λ) e Q-Learning(λ), $\lambda = 0,3$.

5 Conclusão

As curvas que representam a diferença temporal da matriz Q (Figura 3 e 5) indicam a convergência para uma política $\pi^{'}$ qualquer. Além disso, os resultados analisados na forma de gols acumulados não apresentaram indicações de aprendizagem satisfatória. De acordo com as Figuras 4 e 6, é possível observar que as curvas aproximadamente lineares independente do λ utilizado, seguindo uma relação constante entre o número de episódios e o número de gols acumulados. Também, verificou-se um número maior de gols para $\lambda = 0,3$ para ambos os algoritmos.

Comparando o total de gols, verificou-se um melhor desempenho do algoritmo Q-Learning(λ) em relação ao Sarsa(λ) para o modelo discreto, ou seja, seguir a política que fornece o maior valor da

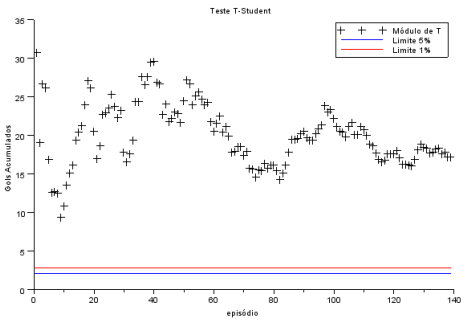

Figura 8: Teste T-Student: Comparação entre Sarsa(λ) e Q-Learning(λ), $\lambda = 0,3$.

função Q, Off-Policy, gerou melhores resultados.

Agradecimentos

Agradecemos à CAPES, CNPQ, FAPEMIG e UFSJ pelo apoio.

Referências

Boer and Kok (2002). *The incremental development of a synthetic multi-agente system: The uva trilearn 2001 robocup soccer simulation team.*, Master's thesis.

F., S., Tizhoosh, H. R. and Salama, M. (2008). Application of reinforcement learning for segmentation of transrectal ultrasound images, *BMC Medical Imaging* **8**: 1–10.

Framling, K. (2007). Replacing eligibility trace for action-value learning with function approximation, *Proceedings - European Symposium on Artificial Neural Networks (ESANN)* pp. 25–27.

Gordon, G. (2001). Reinforcement learning with function approximation converges to a region, *In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), Advances in neural information processing systems* **13**: 1040–1046.

Junior, L. A. C. (2007). *Aprendizado por reforço acelerado por heurísticas no domínio do futebol de robôs simulado.*, Master's thesis, Centro Universitário da FEI.

Kaelbling, L., Littman, M. L. and Moore, W. (1996). Reinforcement learning: A survey, *Journal of Artificial Intelligence Research* **4**: 237–285.

Klopf, A. H. (1972). Brain function and adaptive systems: A heterostatic theory, *Technical Report AFCRL .*: 72–164.

Noda, I. (1995). Soccer server: A simulator of robocup., *Proceedings of AI symposium 95, Japanese Society for Artificial Intelligence* pp. 29–34.

Noda, I. and Stone, P. (2002). The robocup soccer server and cmunited clients: Implemented infrastructure for mas research., *Autonomous Agents and Multi-Agent Systems* .

Sherstov, A. A. and Stone, P. (2005). Function approximation via tile coding: Automating parameter choice, *In J.D. Zucker and I. Saitta, editors, SARA 2005, Lecture Notes in Artificial Intelligence* pp. 194–205.

Silva, A. T. R., Silva, H. G., Santos, E. G., Ferreira, G. B., Santos, T. D. and Silva, V. S. (2010). ibots: Descrição do time, *In Competição Latino Americana de Robótica* .

Simões, M. A. C., Silva, H., Meyer, J., Oliveira, J. D., Cruz, L. and Pessoa, H. (2007). Bahia 2d: Descrição do time, *In XIII Simpósio Brasileiro de Automação Inteligente* .

Stone, P., Sutton, R. S. and Kuhlmann, G. (2005). Reinforcement learning for robocup - soccer keepaway., *Adaptive Behavior* **13(3)**: 165 – 188.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences, *Machine Learning* **3(1)**: 9–44.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction.*, no. 1 edn, Cambridge, MA: MIT Press.

Takahashi, R. H. C. (2007). *Otimização Escalar e Vetorial*. Volume 2: Otimização Escalar.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*, PhD thesis, King's College, Cambridge.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning, *Machine Learning* **8**: 279–292.