APRENDIZADO POR REFORÇO COM GENERALIZAÇÃO DE FUNÇÕES.

Rubisson Duarte Lamperti*, Erivelton Geraldo Nepomuceno*, André 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. The ability of the agent to learn a totally unknown causes complexity of the system increases the time for learning. Therefore, the use of some techniques such as the generalization functions has been used together with RL. The robot soccer simulation 2D has been widely studied due to its characteristic complex and multi-agent. The use of RL in simulation 2D has contributed to the advance in machine learning. In this study we used the Tile Coding in an attempt to reduce learning time in the field of robot soccer 2D, through the generalization of the state space. Q-Learning (λ) Tile and Sarsa (λ) Tile were used to evaluate the reinforcement learning to train attack defense in 2D simulation environment.

Keywords— Reinforcement Learning in Control Systems, robot soccer, Multiagent systems, Tile Coding, Q-Learning, Sarsa.

Resumo— O Aprendizado por Reforço (AR) é uma técnica em que o agente aprende por meio de interação direta com o ambiente. A possibilidade do agente aprender em um ambiente totalmente desconhecido faz com que a complexidade do sistema aumente o tempo para o aprendizado. Por isso, o uso de algumas técnicas como, a Generalização de Funções tem sido utilizada em conjunto com o AR. A simulação de futebol de robôs 2D tem sido amplamente estudada devido a sua característica complexa e multiagente. A utilização do AR na simulação 2D tem contribuído para o avanço no aprendizado de máquinas. Neste trabalho foi utilizado o Tile Coding na tentativa de reduzir o tempo de aprendizado no domínio do futebol de robôs 2d, por meio da generalização do espaço de estados. Os algoritmos Q-Learning(λ) Tile e Sarsa(λ) Tile foram utilizados para avaliar o aprendizado por reforço para o treinamento de ataque contra defesa no ambiente de simulação 2d.

Keywords— Aprendizado por Reforço em sistemas de controle, futebol de robôs, Sistemas Multi-agentes, Tile Coding, Q-Learning, Sarsa.

1 Introdução

O desenvolvimento de máquinas e sistemas inteligentes tem sido o alvo de pesquisadores de diversas partes do mundo (Russell and Norvig, 2004). A Inteligência Artificial (IA) é uma área da ciência que abrange conceitos como, autonomia e inteligência de máquinas. Várias técnicas de IA são adotadas na tentativa de encontrar a melhor solução para uma determinada tarefa, 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, devido às experiências obtidas por meio de tentativa e erro. Por outro lado, a falta de conhecimento prévio do ambiente, a complexidade do sistema e um grande espaço de estados podem contribuir para o aumento do tempo necessário de aprendizado (Sherstov and Stone, 2005).

Na tentativa de minimizar o tempo de aprendizado, diminuindo o espaço de estados, outra técnica como, a Generalização de Funções tem sido utilizada em conjunto com o AR.

A generalização de funções tem sido amplamente utilizada na redução do espaço de estados, como uma alternativa à discretização do espaço, a qual, apesar de fácil implementação dos algoritmos (Sutton and Barto, 1998), pode resultar um mal desempenho no aprendizado, devido a uma discretização ruim (Garcia et al., 2009). Por meio da generalização de função é possível visitar um subconjunto limitado do espaço de estados e produzir uma boa aproximação de todo o espaço (Sutton and Barto, 1998). A generalização de funções é capaz de lidar com variáveis contínuas em várias dimensões, diminuindo o custo computacional e viabilizando o aprendizado de algumas tarefas de maneira eficiente (Bianch, 2004).

Em AR, os ambientes de simulação têm contribuído para o uso da técnica, podendo ser obtidas soluções para o problema antes da aplicação em sistemas reais. Um ambiente que tem sido amplamente estudado e que tem contribuído para o avanço do AR é a simulação de futebol de robôs 2D.

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, por meio de um simulador que fornece um sistema totalmente distribuído, onde, os agentes

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

(ou jogadores) utilizam sensores e atuadores com ruídos, tornando o ambiente incerto (Noda and Stone, 2002).

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 a generalização de funções (Stone et al., 2005).

Sabe-se que o AR apresenta dificuldades de convergência. Essa dificuldade não é geral e não há evidências na literatura que garantam que isso serve para todos os sistemas. Assim, nesse trabalho deseja-se verificar se a aplicação por AR no futebol 2D apresenta também essas características, quando utilizada a generalização de função linear.

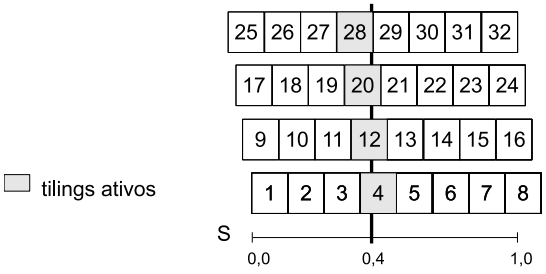

Figura 1: Tile Coding Unidimensional - Generalização do espaço de estados utilizando 4 tilings, totalizando 32 tiles. O vetor de características binário é dado por $\phi(s) = \{0_0, ..., 1_4, 0_5, ..., 1_{12}, 0_{13}, .., 1_{20}, 0_{21}, .., 1_{28}, 0_{29}, .., 0_{32}\}$. Os tiles ativos estão representados pela cor cinza.

2 Fundamentação Teórica

2.1 Generalização de Funções: Tile Coding

Os sistemas modelados por meio de variáveis de estados discretas são facilmente implementados por meio de representações da função valor estado, $V(s)$, ou função valor estado-ação, $Q(s, a)$, na forma tabular.

No entanto, a escolha de uma discretização ruim do espaço de estados pode introduzir estados ocultos no problema, tornando impossível o aprendizado da política ótima (Garcia et al., 2009). Já uma discretização dos estados detalhada perde a capacidade de generalizar, aumentando a quantidade de dados para o aprendizado. Como a maioria dos problemas reais é constituída de variáveis de estados contínuas uma enorme quantidade de memória é exigida para armazenar as tabelas e garantir a precisão. Assim, substituir a discretização de estados por uma representação reduzida, por meio da generalização dos estados, tem sido uma alternativa encontrada capaz de lidar com variáveis contínuas em várias dimensões (Siginam, 2012). Por isso, as técnicas de generalização de função têm sido exaustivamente estudadas e combinadas com AR (Bianch, 2004).

Por meio da generalização de função é possível visitar um subconjunto limitado do espaço de estados e produzir uma boa aproximação de todo o espaço (Sutton and Barto, 1998). Várias abordagens como, Tile Coding (Albus, 1975), também conhecido como CMAC (*Cerebellar Model Articulation Controller*), Redes Neurais (Cetina, 2008) e o Kanerva Coding (Kostiadis and Hu, 2001) têm sido utilizadas com sucesso na aproximação de funções.

O Tile Coding é uma forma de representação grosseira (Stone et al., 2005) que possui as características agrupadas em partições do espaço de estados. Cada partição é chamada de *tiling*, e cada elemento dessa partição é chamado de *tile*, no qual, representa uma característica binária (Garcia et al., 2009). Os *tilings* são sobrepostos e deslocados uns dos outros em uma distância (resolução) r. Na Figura 1, apresenta-se um exemplo de Tile Coding, no qual, representa um espaço de estados unidimensional, S, em que quatro *tilings* foram sobrepostos.

O conjunto de todos os tiles ativos, um por *tiling* e quatro por estado, é representado pelo vetor de característica binário, $\phi(s)$, o qual, pode ser definido por meio da expressão 1.

$$\phi(s) = \begin{cases} 0 & se \quad s \notin tile_t, \\ 1 & se \quad s \in tile_t \end{cases} \tag{1}$$

A combinação linear utilizando a função de mapeamento $\phi(S)$ transforma o estado s em um vetor com N características binárias $[\phi_1, \phi_2, ..., \phi_N]^T$. Assim, a função $Q_a(s)$ é obtida por meio do produto escalar $\phi(s) \times \vec{\theta}$, onde $\vec{\theta}$ é um vetor de dimensão igual a N (Equação 2).

$$\begin{aligned} Q_a(s) &= \vec{\theta_t^T}\vec{\phi_s} \\ Q_a(s) &= \sum_{i=1}^{N} \theta_t(i)\phi_s(i). \end{aligned} \tag{2}$$

A generalização da função Q_a ocorre com a sobreposição dos tilings e, para cada tiling são armazenadas as variáveis que indicam o peso de cada tile. Na equação 2, o vetor $\vec{\theta_t^T}$ representa o conjunto de pesos de cada tile, e $\vec{\phi_s}$ o vetor binário com os tiles ativos.

2.2 Tile Coding e o Aprendizado por Reforço

O Aprendizado por Reforço, por meio dos algoritmos Q-Learning(λ) e Sarsa(λ) quando combinados com o Tile Coding dão origem aos algoritmos de controle Off-Policy (1) e On-Policy (2).

Nestes algoritmos, o vetor de características presentes em (s, a), $\vec{\phi_s}$, é representado pelo vetor binário F_a. A cada episódio, a função Q é obtida por meio da função 2, a qual é representada por

Inicialize $\vec{\theta}$ arbitrariamente ;
para todo *episodio* **faça**
  Inicialize s (estado inicial), a (ação inicial), e, $\vec{e} = \vec{0}$;
  $F_a \leftarrow$ características presentes em s, a ;
  repita
    para todo $i \in F_a$ **faça**
      $e(i) \leftarrow e(i) + 1$;
    fim
    Tome ação a, observe a recompensa r e o próximo estado s' ;
    $\delta \leftarrow r - \sum_{i \in F_a} \theta(i)$;
    para todo $a \in A(s)$ **faça**
      $F_a \leftarrow$ características presentes em s, a ;
      $Q_a \leftarrow \sum_{i \in F_a} \theta(i)$;
    fim
    $\delta \leftarrow \delta + \gamma \max_a Q_a$;
    $\vec{\theta} \leftarrow \vec{\theta} + \alpha\delta\vec{e}$;
    se $(1 - \varepsilon) > rand[0, 1]$ **então**
      para todo $a \in A(s)$ **faça**
        $Q_a \leftarrow \sum_{i \in F_a} \theta(i)$;
      fim
      $a \leftarrow arg\max_a Q_a$;
      $\vec{e} \leftarrow \gamma\lambda\vec{e}$;
      senão
        $a \leftarrow a'$ (escolha aleatória de $a \in A$) ;
        $\vec{e} \leftarrow \vec{0}$;
      fim
    fim
  até *s ser terminal*;
fim

Algoritmo: Q-Learning(λ) Tile utilizando uma política gulosa (ε).

Inicialize $\vec{\theta}$ arbitrariamente ;
para todo *episodio* **faça**
  Inicialize s (estado inicial), a (ação inicial), e $\vec{e} = \vec{0}$;
  $F_a \leftarrow$ características presentes em s, a ;
  repita
    para todo $i \in F_a$ **faça**
      $e(i) \leftarrow e(i) + 1$;
    fim
    Tome ação a, observe a recompensa r e o próximo estado s' ;
    $\delta \leftarrow r - \sum_{i \in F_a} \theta(i)$;
    se $(1 - \varepsilon) > rand[0, 1]$ **então**
      para todo $a \in A(s)$ **faça**
        $F_a \leftarrow$ características presentes em s, a ;
        $Q_a \leftarrow \sum_{i \in F_a} \theta(i)$;
      fim
      $a \leftarrow arg\max_a Q_a$;
      senão
        $a \leftarrow a'$ (escolha aleatória de $a \in A$) ;
        $F_a \leftarrow$ características presentes em s, a ;
        $Q_a \leftarrow \sum_{i \in F_a} \theta(i)$;
      fim
    fim
    $\delta \leftarrow \delta + \gamma Q_a$;
    $\vec{\theta} \leftarrow \vec{\theta} + \alpha\delta\vec{e}$;
    $\vec{e} \leftarrow \gamma\lambda\vec{e}$;
  até *s ser terminal*;
fim

Algoritmo: Sarsa(λ) Tile utilizando uma política gulosa (ε).

$Q_a \leftarrow \sum_{i \in F_a} \theta(i)$ nos algoritmos. A regra de atualização do vetor de características (ou, pesos), $\vec{\theta}$, é dada pela expressão $\vec{\theta} \leftarrow \vec{\theta} + \alpha\delta\vec{e}$, a qual, considera a taxa de aprendizado e o traço de elegibilidade (Peng and Williams, 1994).

2.3 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).

Na plataforma 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. 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

A plataforma 2D da Robocup foi utilizada no ambiente Trainer no modo Off-line. Neste modo foi possível simular a condição de ataque e 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 3).

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

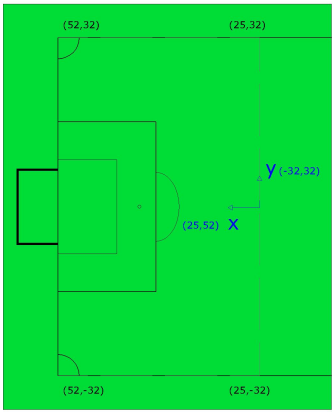

Figura 2: Representação das variáveis de estado contínuas X e Y.

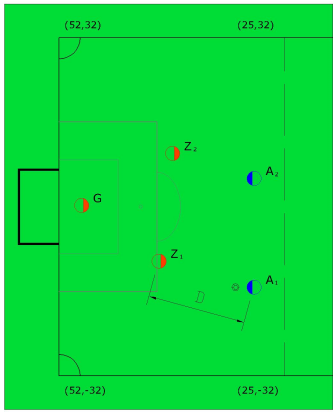

Figura 3: Representação das variáveis de estado - A variável D representa a distância entre o atacante com posse de bola e o defensor mais próximo.

jogadores entre os intervalos $25 \leq X \leq 52$ e $-32 \leq Y \leq 32$, conforme ilustrado na Figura 2. Para cada simulação foram realizadas 140 episódios, sendo cada episódio composto por 12 tentativas a gols (Takahashi, 2007), totalizando 1680 tentativas a gols. 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.

Todas as simulações foram realizadas com os seguintes parâmetros: $\gamma = 0,9$, $\alpha = 0,125$ (Stone et al., 2005), e taxa de exploração igual a 0,05. Os parâmetros γ e exploração são os mesmos utilizados por alguns autores (Junior, 2007; Bianch, 2004; Pegoraro, 2001). O número de tilings foi variado entre 1 e 64, e o número de tiles foi fixado em 4.

Quando necessário foi feita a comparação entre os algoritmos para os valores médios de número de gols acumulados por episódio pelo teste t-student ($p = 1\%$ e $p = 5\%$) (Spiegel, 1984).

A linguagem C++ (compilador GNU g++), baseado no código fonte disponível por Hidehisa Akiyama (Akiyama and Noda, 2008) foi utilizada para a implementação dos algoritmos. As rotinas computacionais foram executadas em um Microcomputador Intel Core i5-2430M @ 2,40 GHz, com 4,0 Gb de memória RAM e sistema operacional Linux.

3.1 Estados Contínuos

As variáveis de estado contínuas foram descritas em relação à posição do agente, por meio da coordenada X e Y, e a variável D que representa a distância entre o agente com posse de bola e o adversário mais próximo. As Figuras 2 e 3 ilustram as variáveis de estados X, Y e D, as quais, são definidas como:

$$S = \begin{cases} 25 \leq X \leq 52, \\ -32 \leq Y \leq 32, \\ 0 \leq D \leq 20. \end{cases} \quad (3)$$

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

As recompensas foram dadas por meio de uma lógica simples, a qual, dependia do gol marcado e a posse de bola. As recompensas foram iguais a -1 para as ações que não resultaram em gol. No caso de gol, a recompensa foi igual a 50. Quando ocorreu a perda de bola para o adversário, o agente recebeu uma recompensa igual a -10.

3.4 Função Valor Generalizada - $Q(s, a)$

A função valor estado-ação $Q_a(s)$ foi estimada segundo os algoritmos Q-Learning(λ) e Sarsa(λ) com generalização de função (Seção 2.2). Neste trabalho, os algoritmos combinados com a generalização de função por meio do Tile Coding são denominados de Q-Learning(λ) Tile e Sarsa(λ) Tile.

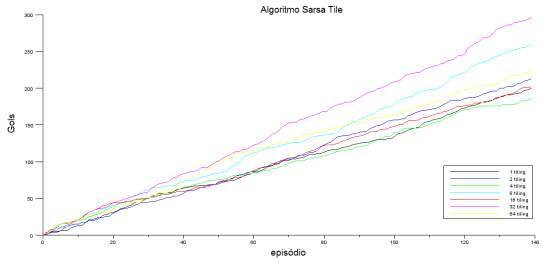

Figura 4: Sarsa - Média dos gols acumulados para $\lambda = 0,3$.

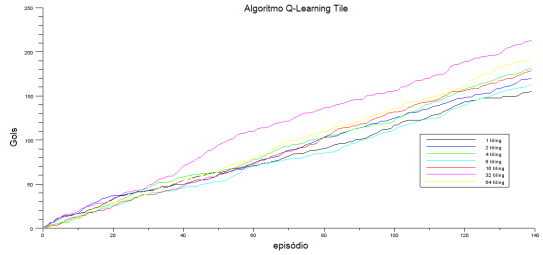

Figura 5: Q-Learning(λ) - Média dos gols acumulados para $\lambda = 0,3$.

4 Análise dos Resultados

4.1 Simulação do Algoritmo Sarsa(λ) Tile

Neste experimento foi observada a relação entre o número de gols acumulados e o número de tilings necessário para a generalização da função $Q_a(s)$ para λ igual a $0,3$. Independente do número de tilings utilizado, houve um aumento crescente no número de gols dentro do intervalo estudado (Figura 4). Além disso, o aumento do número de tilings não teve relação com o aumento do número de gols acumulados. Assim, a simulação com 32 tilings gerou o maior número de gols acumulados (270), seguido da curva com 8 tilings (260). Já as simulações que forneceram menor número de gols acumulados foram com 1 tiling (176) e 4 tilings (160).

4.2 Simulação do Algoritmo Q-Learning(λ) Tile

Variando o número de tilings e mantendo λ igual a $0,3$ no algoritmo Q-Learning(λ) foi possível verificar em todas as curvas um aumento crescente no número de gols durante todo o período de simulação (Figura 5), semelhante ao observado na simulação de Sarsa(λ) Tile. Outra característica comum aos dois algoritmos foi a ausência de relação entre o número de tilings com o aumento do número de gols acumulados. A simulação com maior saldo de gols foi conseguida utilizando 32 tilings (213) e com 1 tiling (155) foi obtido o menor saldo de gols.

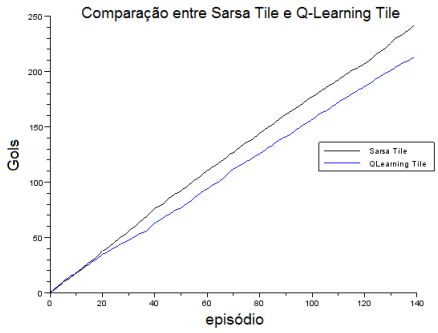

Figura 6: Comparação entre Sarsa Tile(λ) e Q-Learning Tile(λ), $\lambda's = 0,3$.

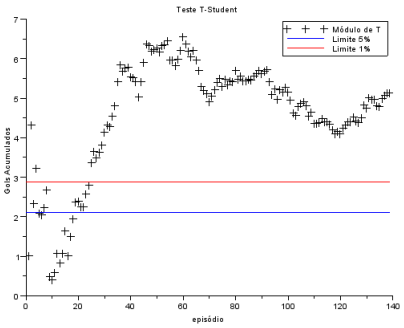

Figura 7: Teste T-Student: Comparação entre Sarsa Tile(λ) e Q-Learning Tile(λ), $\lambda = 0,3$.

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

Os gols marcados durante a simulação do algoritmo Sarsa Tile(λ) e Q-Learning Tile(λ), utilizando λ igual a $0,3$, resultou em um maior número de gols (Figura 6). Ao analisar este gráfico, é possível observar que o algoritmo Sarsa Tile(λ) obteve o maior saldo de gols que Q-Learning Tile(λ), a partir do 28º episódio pelo teste t-student ($p = 1\%$ e $p = 5\%$)(Figura 7).

5 Conclusão

O número de tilings que forneceu melhores resultados foi igual a 32 para ambos os algoritmos. Além disso, não verificou-se relação entre o número de tilings com o aumento do número de gols acumulados para Sarsa(λ) Tile e Q-Learning(λ) Tile.

Um importante resultado obtido pelas simulações foi a prevalência de Sarsa(λ) Tile em relação a Q-Learning(λ) Tile. Este fato, vem confirmar a opinião de alguns pesquisadores, de que, os métodos Off-Policy como Q-Learning(λ) quando utilizados com uma função de aproximação linear podem tornar-se instável (Stone et al., 2005). Já os algoritmos On-policy, não apresentam essa característica de instabilidade quando utilizados em conjunto com uma função de aproximação linear.

Agradecimentos

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

Referências

Akiyama, H. and Noda, I. (2008). Multi-agent positioning mechanism in the dynamic environment, *RoboCup 2007: Robot Soccer World Cup XI* pp. 377–384.

Albus, J. S. (1975). A new approach to manipulator control: The cerebellar model articulation controller (cmac)., *Trans. of the ASME, J. Dynamic Systems, Measurement, and Control* **97**(3): 220–227.

Bianch, R. A. C. (2004). *Uso de Heurística para a Aceleração do Aprendizado por Reforço*, PhD thesis, Escola Politécnica da Universidade de São Paulo.

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

Cetina, V. U. (2008). Multilayer perceptrons with radial basis functions as value functions in reinforcement learning, *European Symposium on Artificial Neural Networks - Avances in Computational Intelligence and Learning (ESANN'2008)* pp. 23–25.

Garcia, J., Bueno, I. L., Fernandez, F. and Borrajo, D. (2009). *A Comparative Study of Discretization Approaches*, Computer Science Department, Universidad Carlos III de Madrid Avenida de la Universidad 30, 28911 Leganes, Madrid, Spain, chapter A Comparative Study of Discretization Approaches for State Space Generalization in the Keepaway Soccer Task, pp. 1 – 31.

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.

Kostiadis, K. and Hu, H. (2001). Kabage - rl: Kanerva - based generalization and reinforcement learning for possession football, *Proceedings of IEEE/RSJ International Conference on Intelligente Robots and Systems* .

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* .

Pegoraro, R. (2001). *Agilizando Aprendizagem por Reforço em Robótica Móvel Através do Uso de Conhecimento Sobre o Domínio.*, PhD thesis, EPUSP/PCS.

Peng, J. and Williams, R. J. (1994). Incremental multi-step q-learning., *Machine Learning: Proceedings of the Eleventh International Conference* .: 226–232.

Russell, S. J. and Norvig, P. (2004). *Inteligência Artificial*.

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.

Siginam, B. (2012). *Adaptive Tile Coding Methods for the generalization of value functions in the RL state space.*, PhD thesis, University of Mminnesota.

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* .

Spiegel, M. R. (1984). *Estatística*, 2 edn, São Paulo: McGraw-Hill.

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.