6
REFERÊNCIAS BIBLIOGRÁFICAS

ARVO, J. **Linear Time Voxel Walking for Octrees**. Ray Tracing News, Feb 1988.

BARTO, A. G. **Reinforcement Learning and Adaptive Critic Methods**, Handbook Of Intelligent Control: Neural, Fuzzy, And Adaptive Approaches, NY, White, D.A.; Sofge, D.A. (eds), Van Nostrand, Reinhold, 1992. p.469-491.

BONARINI, A. **Fuzzy Sets and Evolutionary Learning: Motivations and Issues for Integrated Systems**. Proceedings of The Icml '96 Pre-Conference Workshop on Evolutionary Computing and Machine Learning, University of Bari, Bari, I, p.30-37, 1996.

BOUTILIER, C. **Planning, Learning and Coordination in Multiagent Decision Process**. TARK, 1996.

BOYAN, J. A.; MOORE, A. W. **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, 1995.

BRENNER, W.; ZARNEKOW, R.; WITTIG, H. **Intelligent Software Agents**, Springer, 1998.

BROWN M.; BOSSLEY K. M.; MILLS D. J.; HARRIS C. J. **High Dimensional Neurofuzzy Fystems: Overcoming the Curse of Dimensionality**. Proceedings. Int Joint Conf. On Fuzzy Systems And The 2nd Int. Fuzzy Engineering Symp, Yokohama, Japan, March 1995, p.2139-2146.

DUDEK, G.; JENKIN, M. **Computational Principals of Mobile Robotics**. Cambridge University Press, USA, 2000.

FIGUEIREDO, K. **Novos Modelos Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço para Agentes Inteligentes**. Tese de Doutorado, Departamento de Engenharia Pontifícia Universidade Católica do Rio de Janeiro, 2003.

FIGUEIREDO, K.; VELLASCO, M.; PACHECO, M. A. C.; SOUZA, F. J. **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, 2005, pp. 4-13.

FIGUEIREDO, K.; VELLASCO, M.; PACHECO, M. A. C.; SOUZA, F. J. **Reinforcement Learning-Hierarchical Neuro-Fuzzy Politree Model for Control of Autonomous Agents**. Fourth Inter-national Conference on Hybrid Intelligent Systems (HIS'04), pp.130-135, (ISBN 0-7695-2291-2), IEEE Comp. Society, Japan, Dec 2004.

FINKEL, R. A.; BENTLEY J. L. **Quad-Trees , a Data Structure For Retrieval On Composite Keys**. Acta Informatica 4, 1974, p. 1-9.

FLESCH, G. G. **Policy Improvement for the Reinforcement Learning - Hierarchical Neuro Fuzzy Politree Model (RL-HNFP)**. Master Thesis of University of Duisburg-Essen Chair of Dynamics and Control, 2009.

FRÄMLING, K. **Light-weight reinforcement learning with function approximation for real-life control tasks**. Proceedings of 5th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Funchal, Madeira, Portugal, 11-15 May, 2008. pp. 127-134.

GLORENNEC, P. Y.; JOUFFE L. **Fuzzy Q-Learning**. Proceedings Of Fuzz-Ieee'97, Sixth International Conference on Fuzzy Systems, Barcelone, Espagne, Juillet 1997, p. 659-662.

GORDON, G. J. **Stable function approximation in dynamic programming**. Armand Prieditis and Stuart Russell, editors, Proceedings of the Twelfth International Conference on Machine Learning, San Francisco, CA,.Morgan Kaufmann,1995.

HAYKIN S. **Neural Networks – A Comprehensive Foundation**. Macmillan College Publishing Company, Inc, 1998.

JANG, J. S. R. **ANFIS: Adaptive-Network-Based Fuzzy Inference System**. Ieee Transactions on Systems, Man, and Cybernetics, Vol.23, N.3, may/june 1993, p.665-685.

JENNING, N. R.; WOOLDRIDGE, M. J. **Agent Technology: Fundations, Applications, and Markets**, Springer, December, 1997.

JONG, N. K.; STONE, P. **Kernel-Based Models for Reinforcement Learning**. ICML-06 Workshop on Kernel Methods and Reinforcement Learning, Pittsburgh, PA, June 2006.

JOUFFE, L. **Fuzzy Inference System Learning by Reinforcement Methods**. Ieee Transactions on Systems, Man and Cybernetics, part c, vol.28, n. 3, 1998, p.338-355.

KAELBLING, L. P.; LITTMAN M. L.; MOORE, W. A. **Reinforcement Learning: A Survey**. Journal of Artificial Intelligence Research 4, May 1996, p. 237-285.

KENDALL, E. A. PATHAK, C. V. KRISHNA, P. V. M. SURESH, C. B. **The Layered Agent Pattern Language**. Proceedings of the Conference on Pattern Languages of Programs, 1997.

KRUSE, R.; NAUCK, D. **NEFCLASS-A Neuro-Fuzzy Approach for the Classification of Data**. Proceedings. of the Acm Symposium on Applied Computing, Nashville, 1995.

LIN, L. J. **Self-improving reative agents based on reinforcement learning, planning and teaching**. Machine Learning, vol. 8, no.3/4, May 1992, p. 293-321.

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

MILLER, W. T.; GLANZ, F. H.; KRAFT, L. G. **CMAC: An Associative neural network alternative to backpropagation**. Special Issue on Neural Networks, II, vol. 78, October 1990, p. 1561-1567.

MOORE, A. W. **Variable resolution dynamic programming:Efficiently learning action maps in multivariate real-valued state-spaces**. Proceedings of the Eighth International Conference on Machine Learning, L.A., Birnbaum and G.C.Collins, eds., Morgan Kaufmann, 1991, p. 333-337.

NAUCK, D.; KRUSE, R. **NEFCON-I: An X-Windows based simulator for Neural-Fuzzy Controllers**. Proceedings Ieee International Conference Of Neural Network, at IEEE WCCI´94, Orlando, 1994, p. 1638-1643.

NÜRNBERG, A.; NAUCK, D.; KRUSE, R**. Neuro-Fuzzy Control Based on the NEFCON-model: recent developments**. Soft Computing, Spring-Verlag, 1999, p. 168-182.

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

PYATT, L. D.; HOWE, A. H. **Decision Tree Function Approximation in Reinforcement Learning**. Technical Reports 98-112, Computer Science Department, Colorado State University, 1998.

RIBEIRO, C. H. C. **A Tutorial on Reinforcement Learning Techniques**. International Joint Conference on Neural Networks ed. : INNS Press, 1999.

SINGH, S. P.; SUTTON, R. S. **Reinforcement learning with replacing eligibility traces**. Machine Learning, vol. 22, no. 1, 1996, p. 123-158.

SOUZA, F. J. Modelos **Neuro-Fuzzy Hierárquicos**. Tese de Doutorado, Departamento de Engenharia Elétrica Pontifícia Universidade Católica do Rio de Janeiro, Abril 1999.

SOUZA, F. J.; VELLASCO, M. M. B. R.; PACHECO, M. A. C. **Hierarchical Neuro-Fuzzy QuadTree Models**. Fuzzy Sets & Systems, ISSN 0165-0114, vol 130/2, August 2002, Elsevier Science, Amsterdam, The Netherlands, p 189-205.

SUN, R.; SESSIONS C. **Self-Segmentation of Sequences: Automatic Formation of Hierarchies of Sequencial Behaviours**. IEEE Transactions on Systems, Man and Cybernetics: Part B, Cybernetics, Vol.30, 2000, p.403-418.

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

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. MIT Press, Cambridge, MA, 1998.

TAMMINEN, M. **Comment on quad- and octtrees**. Communications of the ACM, 27(3), 1984, p. 248-249.

UTHER, W. T. B.; VELOSO, M. M. **Tree based discretization for continuous state space reinforcement learning**. Proceedings of AAAI-98, Madison, WI, July 1998.

VUORIMAA, P. **Fuzzy self-organing map**. Fuzzy Sets and Systems, No.66, 1994, p.223-231.

WATKINS, C. J. C. H. **Learning from Delayed Rewards**. Ph.D. thesis, University of Cambridge, England, 1989.

Apêndice
APRENDIZADO POR REFORÇO

O aprendizado por reforço, ou *reinforcement learning* (RL) (Barto, 1992), pode ser usado para aprender a controlar um processo, onde, uma vez aplicado a agentes autônomos, estes poderão obter o comportamento desejado. Este algoritmo, baseado na iteração do agente com o ambiente, recebe do ambiente um reforço, que é uma avaliação do desempenho atual. Através deste reforço, o algoritmo modifica o controlador de modo que este selecione ações que melhor implementem o comportamento esperado.

Esta forma de aprendizagem é inspirada na maneira de animais e seres humanos aprenderem determinados comportamentos a partir de estímulos (recompensa ou punição) que recebem do ambiente. O método de aprendizado por reforço tenta aplicar este conceito na resolução de problemas principalmente relacionados a área de controle. O principal benefício é que o sistema que utiliza aprendizado por reforço pode aprender como resolver uma tarefa autonomamente, sem qualquer conhecimento prévio do sistema.

O agente conhece o conjunto de ações que pode realizar para atuar em seu ambiente. Também pode conhecer algo sobre o ambiente, como um modelo simplificado ou completo, mas isso não é obrigatório. Para cada ação executada pelo agente, através de atuadores, o ambiente realimenta o agente, por meio de uma nova leitura dos sensores, com um novo estado e atribui uma recompensa para a ação realizada naquele estado. A recompensa pode ser qualquer valor escalar que indique ao agente se a última ação foi boa ou ruim. Uma abordagem intuitiva é atribuir valores negativos de recompensa para maus resultados, punindo-os, e valores positivos para bons resultados, incentivando-os. Quanto maior o valor da recompensa, melhor é o resultado da escolha daquela ação. Ações que resultam em alta recompensa possuem mais chances de serem escolhidas no futuro em situações semelhantes.

Como mostra a figura A.1, o sinal de recompensa não faz parte dos conhecimentos do agente. Tecnicamente, o sinal de recompensa é obtido partir de informações relacionadas ao estado ou meta do agente.

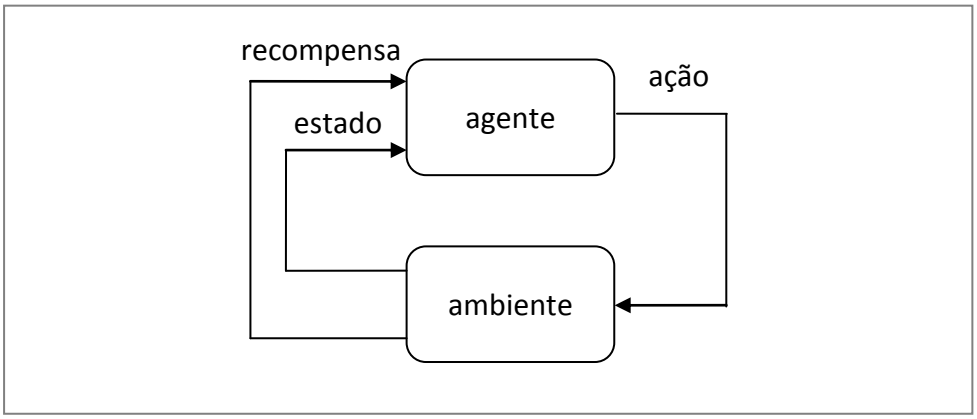

Figura A.1: Algoritmo de Aprendizado por Reforço.

A decisão sobre qual ação será executada é realizada através de uma política de escolha. Há diferentes tipos de abordagens para a criação de uma política, todas intimamente relacionadas aos algoritmos de aprendizado. Algoritmos de aprendizado podem ser encontrados em programação dinâmica, métodos de Monte Carlo e aprendizado por diferença temporal (Sutton & Barto, 1998).

Os seres humanos são capazes de prever se uma dada ação que parece ser boa à primeira vista leva a resultados ruins no futuro e, portanto, evitá-la. Por outro lado, uma ação que parece ruim pode levar a recompensas boas no futuro. Se o agente for capaz de simular este comportamento, seria capaz de aprender com as suas ações passadas e aplicar esse conhecimento para o futuro. Processos de decisão de Markov (Sutton & Barto, 1998) são utilizados para formular este modelo de recompensa baseado na acumulação de informações passadas.

A.1

Processo de decisão de Markov

A recompensa, ou reforço, é a chave do aprendizado RL. A única informação que um agente tem sobre o que deve fazer, seu objetivo, é dado pelos valores de recompensa. Estes valores induzem o agente a selecionar suas ações, porém não mostram como realizar seu objetivo. Este deve ser alcançado por meio de tentativa e erro. Um agente busca maximizar o retorno total esperado, dado pela soma das recompensas de todos os passos até o momento no qual o agente atinge o seu objetivo no instante T.

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T \tag{A.1}$$

Esta abordagem é denominada como modelo de horizonte finito (Kaelbling et al., 1996), pois existe um número finito de passos. Após alcançar seu objetivo, inicia-se um novo episódio no qual o agente tem a oportunidade de melhorar o seu conhecimento. Este tipo de problema finito consiste em inúmeros episódios, onde o agente deve resolver o problema diversas vezes melhorando a cada tentativa, como um aprendiz.

Em alguns casos, como nos problemas de ambiente contínuo que são tratados neste trabalho, se têm um número muito grande de passos e não se sabe de antemão quantos são (não é um número fixo). Nestes casos, outro modelo, chamado de horizonte infinito deve ser usado. Uma taxa de desconto γ reduz o impacto dos retornos que estão mais distantes do retorno:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \tag{A.2}$$

Em outras palavras, apenas a recompensa da próxima etapa é considerada plenamente. O valor γ pode ser interpretado, por exemplo, como uma taxa de juros, onde passos distantes ainda são multiplicados por um fator pequeno, não sendo assim muito importantes para o agente. O valor de desconto γ varia de ‘0’ a ‘1’. Caso γ se aproxima de ‘1’, o agente valoriza todas as futuras recompensas, mesmo comportamento do modelo de horizonte finito. Se γ se aproxima de ‘0’, o agente só leva em conta a recompensa da próxima ação, tornando-se um agente míope. A função de recompensa para calcular R depende do problema a ser resolvido pelo agente.

A fim de determinar recompensas futuras, uma formulação para descrever as ações e estados é necessária, porque a execução de uma ação não só resulta em um sinal de recompensa, mas também pode alterar o estado do agente. Um processo de decisão de Markov (MDP) (Sutton & Barto, 1998) descreve como os estados mudam quando o agente executa certas ações no ambiente, fato de suma importância no aprendizado por reforço. Ambos os modelos de horizonte finito e infinito podem ser descritos usando MDP's.

Um estado é definido a partir das informações completas que o agente possui. Estas se baseiam na base de conhecimento do agente obtida por meio das leituras dos sensores.

As ações descrevem o que o agente pode realizar no ambiente através de seus atuadores. Por exemplo, um robô pode se movimentar para frente, para trás e rodar em torno de seu eixo, em diferentes velocidades.

Para a aprendizagem por reforço, MDP's finitos são mais importantes. O MDP é finito, quando atende as propriedades de Markov. Neste caso, o estado contém informações suficientes para recuperar as informações necessárias dos estados passados. Na realidade isto não ocorre sempre, mesmo assim esta suposição é boa o suficiente.

Um exemplo de um MDP finito é um jogo de xadrez. Um estado em um jogo de xadrez pode ser representado pela posição de todas as peças no jogo. Para o cálculo da próxima ação, não importa de que posição as peças vieram. Este problema depende apenas do estado atual, em outras palavras, o estado atual contém toda a informação necessária.

A probabilidade de cada possível estado seguinte s' depende do estado atual s, a ação adotada a e da dinâmica da transição dos estados. $P^a_{ss'}$ é chamada de probabilidade de transição.

$$P^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\} \tag{A.3}$$

A recompensa $R^a_{ss'}$ é calculada de forma semelhante:

$$R^a_{ss'} = E\{r_{t+1} = r \mid s_t = s, a_t = a, s_{t+1} = s'\} \tag{A.4}$$

Esta função descreve a recompensa esperada após se executar uma ação a no estado s e migrar para o estado s' recebendo o retorno r.

A.3

Função de Valor

A função de valor quantifica a importância de determinado um estado, obtido por meio dos sensores, na conclusão de um objetivo. Ela diz ao agente, quão vantajoso cada estado é. A função valor de estado representa um modelo abstrato do ambiente:

$$V^*(s) = E_\pi\{R_t \mid s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+1+k} \mid s_t = s\right\} \quad \text{(A.5)}$$

Além disso, é possível definir um valor para cada ação que pode ser adotada em cada estado. A função de valor do par estado-ação é uma boa representação de quão boa a escolha pelo agente de determinada ação é em um dado estado.

$$Q^*(s,a) = E_\pi\{R_t \mid s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+1+k} \mid s_t = s, a_t = a\right\} \quad \text{(A.6)}$$

No início, o agente não tem conhecimento sobre o ambiente, e as funções de valor ótimas V*(s) e Q*(s,*a*) não são conhecidos. Eles devem ser estimados durante o processo de aprendizagem. Isto é feito escolhendo, através de uma política de seleção de ação π, e executando as ações nos estados, ou seja, explorando o ambiente e testando o sucesso das ações em cada estado. Usando o sinal de recompensa, o agente é capaz de estimar funções de valor V(s) e Q(s,*a*), que convergem para V*(s) e Q*(s,*a*), respectivamente, se todas as ações em todos os estados forem tomadas um número adequado de vezes.

Adotando cada ação em cada estado apenas uma vez, não é o suficiente, pois, em geral, a função de recompensa ótima R*(s,a) não é conhecida. No modelo de horizonte infinito, o cálculo da recompensa R_t pressupõe conhecimento sobre recompensas futuras, que devem ser estimativas, também. Logo, o agente tem que usar uma média do sinal de recompensa que ele recebe para cada ação em cada estado, e assim, utilizá-lo mais de uma vez para aproximar as funções de valor V*(s) e Q*(s,*a*). Este tipo de método é chamado Método de Monte Carlo.

A.4

Exploração vs. Aproveitamento

A política de escolha de um agente define qual ação deve ser adotada em determinado estado. Deve-se proporcionar um equilíbrio entre a exploração (*exploration)* e o aproveitamento (*exploitation*). O sentido do aproveitamento é executar a ação que a agente espera obter a maior recompensa imediata ou em longo prazo. A princípio, como inicialmente o agente não tem nenhum conhecimento pré-definido, ele pode ainda não ter descoberto a melhor ação possível. Devido a este fato, exploração é essencial: a fim de tentar encontrar outras ações que sejam melhores que as boas ações conhecidas. Na escolha das melhores ações, deve-se considerar o reforço a longo prazo, ao invés do recompensa imediata.

Kaelbing (1996) divide as técnicas disponíveis em técnicas formalmente justificadas e técnicas *ad-hoc*. Os primeiros são bem desenvolvidos, com provas matemáticas de convergência, enquanto os últimos são essencialmente desenvolvidos heuristicamente. Como exemplos de técnicas formalmente justificadas tem-se o método de programação dinâmica. Estes métodos possuem a grande desvantagem de não poderem ser usados em ambientes complexos, porque o número de entradas e os estados se tornam demasiado grandes, como em problemas contínuos com um número infinito de estados. Além disso, não existe garantia de convergência em problemas contínuos.

Como a maioria dos problemas do mundo real são contínuos ou podem conter um grande número de estados, as técnicas *ad-hoc* tornam-se uma melhor escolha. A título de exemplo tem-se a política de escolha *ε-greedy*. De acordo com esta política a melhor ação possível no estado atual, maior função valor Q(s,*a*), será adotada com uma probabilidade 1- ε. Supondo um valor de partida de ε = 0,1, significa que em 90% dos casos a melhor ação conhecida é selecionada (aproveitamento), enquanto em 10% dos casos, uma ação aleatória é selecionada, significando explorar o retorno de outras ações.

Apesar do fato de que as técnicas formalmente justificadas têm garantias de convergência através de provas matemáticas.

A.5

Algoritmos de aprendizado

É tarefa do algoritmo de aprendizagem encontrar uma política ótima para um dado problema. Quando o modelo do ambiente é conhecido, métodos como programação dinâmica (DP) podem ser usados a fim de determinar as funções de valor V*(s) e Q*(s,*a*), diretamente. Caso não exista um modelo, métodos Monte Carlo (MC) podem ser utilizados para explorar o ambiente e melhorar a estimativa das funções de valor após cada episódio.

Os algoritmos de aprendizado utilizados neste trabalho são métodos de diferença temporal (TD). Para estes, nenhum modelo é necessário, e eles podem atualizar as funções de valor após cada passo do agente no ambiente. Isto é essencial em um ambiente contínuo, com tarefas não episódicas. Os valores de Q(s,*a*) são atualizados após uma diferença temporal específica, que é dada por certo número de passos. A versão mais simples do TD é TD (0), que atualiza os valores de Q(s,*a*) a cada passo. Uma forma mais geral deste método é o TD, onde os valores de Q(s,*a*) são atualizados levando em conta estados passados, *elegibility trace*.

Uma desvantagem do TD é que o processo de aprendizagem da política torna-se misto com a execução da política. Para resolver este problema, dois tipos diferentes métodos TD são definidos: Em métodos *on-policy*, a seleção da ação possui influência direta na atualização das funções de valor, enquanto que os métodos *off-policy* são independentes da execução da ação atual.

A.5.1

SARSA

SARSA (Sutton & Barto, 1998) significa *state-action-reward-state-action* que são as variáveis necessárias para realizar a aprendizagem: o estado atual, a ação escolhida neste estado, o próximo estado, a recompensa obtida pela escolha daquela ação e a próxima ação que será adotada. É um algoritmo de controle *on-policy* que aprende a função de valor Q*(s,*a*), ao invés da V*(s).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \quad \text{(A.7)}$$

O valor de Q depende do antigo valor de $Q(s_t,a_t)$ naquele estado para aquela ação, da taxa de aprendizagem $\alpha \in [0,1]$, do reforço r_{t+1}, do fator de desconto $\gamma \in [0,1]$, e do valor de $Q(s_{t+1},a_{t+1})$ da ação que foi escolhido pela política no próximo passo.

Este algoritmo é executado a cada passo do agente no ambiente. Segundo o método TD(0), os valores de Q_t são atualizados em t+1, usando o retorno observado r_{t+1} e o valor de Q_{t+1}.

A.5.2

Q-Learning

O *Q-Learning* desenvolvido por Watkins (1989) é um dos mais utilizados algoritmos de aprendizagem. Ele é semelhante ao SARSA, porém *off-policy*:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \quad \text{(A.8)}$$

A função de valor de Q é atualizada por meio de seu antigo valor $Q(s_t,a_t)$ naquele estado para aquela ação, da taxa de aprendizagem $\alpha \in [0,1]$, do reforço r_{t+1}, do fator de desconto $\gamma \in [0,1]$, e do maior valor de $Q(s_{t+1},a_{t+1})$ no próximo estado.

A simplicidade do algoritmo *Q-Learning* permite que a política e o algoritmo de aprendizagem possam ser avaliados independentemente.

A.5.3

Elegibility Trace

Os rastros de elegibilidade (Singh & Sutton, 1996) podem ser definidos como intermediários entre os métodos de Diferenças Temporais e o método de Monte Carlo, isto é, este método exige mais do que estados imediatos (para o caso de TD) e exige menos quando não prescinde que o sistema alcance o estado final

(caso Monte Carlo). Assim, quando são inseridos em um método TD qualquer, é produzida uma família de métodos que varrem um espectro no qual um dos extremos é o método de Monte Carlo, e o outro, os métodos de Diferenças Temporais de apenas um passo.

O método intermediário TD(λ) resultante entre aqueles dois, com visões intermediárias dos estados, são quase sempre mais eficientes do que os métodos extremos. Neste sentido, a utilização dos rastros de elegibilidade unifica os métodos de Monte Carlo e Diferenças Temporais, de uma forma valiosa e interessante (Sutton & Barto, 1998).

Numa visão mais prática, os rastros de elegibilidade são registros temporários da ocorrência de algum evento, como a visita de um estado ou a seleção de uma ação. Assim, este parâmetro gera um valor de memória associado com os eventos elegíveis para as mudanças posteriores do aprendizado.

No método TD(λ), o eco no passado faz com que a ação n passos a frente influencie os passos antigos. O *eligibility trace* para o estado s no tempo t é definido por $e_t(s,a)$ e representa o número de visitas que este estado recebeu em um passado recente. A cada iteração os *eligibility trace* de todos os estados são reduzidos em $\gamma\lambda$, e o *eligibility trace* do estado visitado é incrementado em 1 (Sutton & Barto, 1998).

$$e_t(s,a) = \begin{cases} \lambda\gamma e_{t-1}(s,a) & se\ s \neq s_t\ ou\ a \neq a_t \\ \lambda\gamma e_{t-1}(s,a) + 1 & se\ s = s_t\ e\ a = a_t \end{cases} \qquad \text{(A.9)}$$

Onde γ é o fator de desconto e $\lambda \in [0,1]$ é o parâmetro que indica o fator de influência dos estados anteriores.

A função de valor é atualizada de forma similar ao mostrado anteriormente, para o caso do SARSA a equação de atualização se torna:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\, e_t(s_t,a_t)\left[r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right] \qquad \text{(A.10)}$$

Este tipo de atualização requer maior processamento computacional, mas a velocidade de convergência é, em geral, maior. Existem algumas restrições ao uso de *eligibility trace* como, por exemplo, quando o algoritmo utiliza excessivamente políticas de escolha de ações *non-greedy* (Sun & Sessions, 2000).